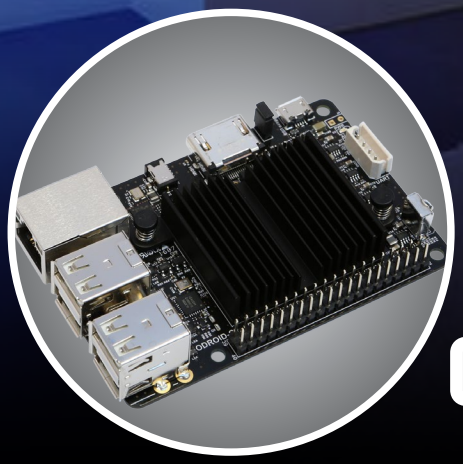
Juegos Android • Sensores de Imagen Avanzado • Pila Android Wifi

ODROID Magazine

Año Tres
Num. #35
Nov 2016



Ambilight

Ultra-HD 4K ODROID

Crea una pantalla con iluminación y video sincronizado

- Juegos Linux: Lograr ejecutar Serious con Serious-Engine
- Android Nougat: Java badado en OpenJDK y nueva API de gráficos

Qué defendemos...

Nos esmeramos en presentar una tecnología punta, futura, joven, técnica y para la sociedad de hoy.

Nuestra filosofía se basa en los desarrolladores. Continuamente nos esforzamos por mantener estrechas relaciones con éstos en todo el mundo.

Por eso, siempre podrás confiar en la calidad y experiencia que representa la marca distintiva de nuestros productos.

Simple, moderno y único.
De modo que tienes a tu alcance lo mejor para hacer todo lo que imagines

HARDKERNEL

Pollin ELECTRONIC

Realizamos envios de ODROID-C2 and ODROID-XU4 a los países de la UE! Ven y visita nuestra tienda online!

Dirección: Max-Pollin-Straße 1
85104 Pförring Alemania

Teléfono & Fax
telf : +49 (0) 8403 / 920-920
email : service@pollin.de

Nuestros productos ODROID se pueden encontrar en: http://bit.ly/1tXPXwe

EDITORIAL

Uno de los periféricos más versátiles para el ODROID es Arduino, que puede ser programado como un controlador independiente para muchos proyectos, desde robots hasta domótica. Un simple proyecto para empezar con Arduino es crear un sistema Ambilight, que es un impresionante sistema de luces de fondo sincronizado con el video que se reproduce en una pantalla. Los ingenieros de Hardkernel lo demostraron en la ARM TechCon 2016, y han escrito una guía para que puedas crear fácilmente el mismo escenario de luces en tu propia casa.

Para mejorar aún más tu experiencia multimedia, presentamos un tutorial sobre cómo configurar el front-end de MythTV, así como un artículo sobre cómo activar la reproducción de vídeo acelerado en un navegador web en el ODROID-C2. Para los entusiastas más experimentados del bricolaje, Miltiadis presenta su proyecto de notificación SMS con controlador de luces que se puede adaptar y ampliar a cualquier aplicación IoT, y Jörg nos muestra cómo configurar un sistema de alarma con sensores de ventanas. Tobias nos introduce en el motor de juego Serious, Nanik describe la pila WiFi de Android y Bruno se divierte con Ancestor, un impresionante juego para Android con una sorprendente jugabilidad.

ODROID Magazine, que se publica mensualmente en http://magazine.odroid.com/, es la fuente de todas las cosas ODROIDianas. • Hard Kernel, Ltd. • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815 • fabricantes de la familia ODROID de placas de desarrollo quad-core y la primera arquitectura ARM "big.LITTLE" del mundo basada en una única placa.
Para información sobre cómo enviar articulos, contacta con odroidmagazine@gmail.com, o visita http://bit.ly/1ypImXs.
Únete a la comunidad ODROID con miembros en más de 135 países en http://forum.odroid.com/ y explora las nuevas tecnologías que te ofrece Hardkernel en http://www.hardkernel.com/

HARDKERNEL

NUESTRO MARAVILLOSO PRESONAL ODROIDIAN:

Rob Roy, Editor Jefe

Soy un programador informático que vive y trabaja en San Francisco, CA, en el diseño y desarrollo de aplicaciones web para clients locales sobre mi cluster ODROID. Mis principales lenguajes son jQuery, angular JS y HTML5/CSS3. También desarrollo SO precompilados, Kernels personalizados y aplicaciones optimizadas para ODROID basadas en las versiones oficiales de Hardkernel, por los cuales he ganado varios Premios. Utilizo mi ODROIDs para diversos fines, como centro multimedia, servidor web, desarrollo de aplicaciones, estación de trabajo y como plataforma de juegos. Puedes echar un vistazo a mi colección de 100 GB de software ODROID, kernel precompilados e imágenes en `http://bit.ly/1fsaXQs`.

Bruno Doiche, Editor Artístico Senior

Recientemente Bruno hace chanchullos con 2 de sus ODROIDs, jugando a juegos y asombrándose por la capacidad de respuesta de sus máquinas con este nuevo y sorprendente sistema. Asegura que nunca se queda sin ideas para la columna de juegos en la que los lectores descubren nuevos juegos junto a él.

Manuel Adamuz, Editor Español

Tengo 31 años y vivo en Sevilla, España, aunque nací en Granada. Estoy casado con una mujer maravillosa y tengo un hijo. Hace unos años trabajé como técnico informático y programador, pero mi trabajo actual está relacionado con la gestión de calidad y las tecnologías de la información: ISO 9001, ISO 27001, ISO 20000 Soy un apasionado de la informática, especialmente de los microordenadores como el ODROID, Raspberry Pi, etc. Me encanta experimentar con estos equipos y traducir ODROID Magazine. Mi esposa dice que estoy loco porque sólo pienso en ODROID. Mi otra afición es la bicicleta de montaña, a veces participo en competiciones semiprofesionales.

Nicole Scott, Editor Artístico

Soy una experta en Producción Transmedia y Estrategia Digital especializa en la optimización online y estrategias de marketing, administración de medios sociales y producción multimedia impresa, web, vídeo y cine. Gestiono múltiples cuentas con agencias y productores de cine, desde Analytics y Adwords a la edición de vídeo y maquetación DVD. Tengo un ODROID-U3 que utilizo para ejecutar un servidor web sandbox. Vivo en el área de la Bahía de California, y disfruta haciendo senderismo, acampada y tocando música. Visita mi web http://www.nicolecscott.com.

James LeFevour, Editor Artístico

Soy un especialista en medios digitales que disfruta trabajando como freelance en marketing de redes sociales y administración de sitios web. Cuanto más aprendo sobre las posibilidades de ODROID más me ilusiona probar cosas nuevas con él. Me traslade a San Diego desde el Medio Oeste de los EE.UU. Continuo muy enamorado de muchos de los aspectos que la mayoría de la gente de la Costa Oeste ya da por sentado. Vivo con mi encantadora esposa y nuestro adorable conejo mascota; el cual mantiene mis libros y material informático en constante peligro.

Andrew Ruggeri, Editor Adjunto

Soy un ingeniero de sistemas Biomédicos anclado en Nueva Inglaterra que actualmente trabaja en la industria aeroespacial. Un microcontrolador 68HC11 de 8 bits y el código ensamblador son todo lo que me interesa de los sistemas embebidos. Hoy en día, la mayoría de los proyectos en los que trabajo están en lenguajes C y C ++, o en lenguajes de alto nivel como C# y Java. Para muchos proyectos, utilizo placas ODROID, pero aún sigo intentando utilizar los controladores de 8 bits cada vez que puedo (soy un fan de ATMEL). Aparte de la electrónica, soy un amante de la fotografía analógica y desarrollo la película friki con la que disfruto intentando hablar en idiomas extranjeros.

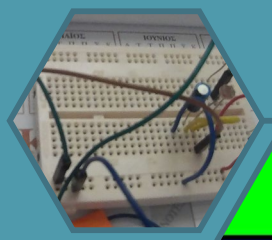
Venkat Bommakanti, Editor Adjunto

Soy un apasionado de los ordenadores desde la bahía de San Francisco en California. Procuro incorporar muchos de mis intereses en proyectos con ordenadores de placa reducida, tales como pequeños modificaciones de hardware, carpintería, reutilización de materiales, desarrollo de software y creación de grabaciones musicales de aficionados. Me encanta aprender continuamente cosas nuevas, y trato de compartir mi alegría y entusiasmo con la comunidad.

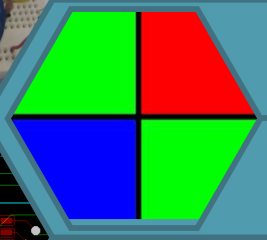
Josh Sherman, Editor Adjunto

Soy de la zona de Nueva York, y ofrezco mi tiempo como escritor y editor para ODROID Magazine. Suelo experimentar con los ordenadores de todas las formas y tamaños: haciendo trizas las tablets, convirtiendo Raspberry Pi en PlayStations y experimentado con los ODROIDs y otros SoCs. Me encanta trabajar con los elementos básicos y así poder aprender más, y disfrutar enseñando a otros escribiendo historias y guías sobre Linux, ARM y otros proyectos experimentales divertidos.

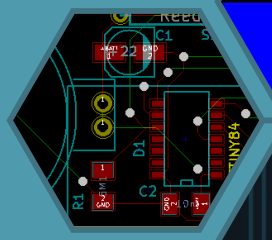
INDICE



DISPOSITIVO IOT - 6



SENSORES DE IMAGENES AVANZADOS - 15



CENTRAL DE ALARMAS - 17

ANCENSTOR - 22

AMBILIGHT - 23

DOCKER - 26

JUEGOS LINUX - 31

DESARROLLO ANDROID - 33

MYTH TV - 36

ANDROID NOUGAT - 39

AYUDA CON EL VIDEO - 40

CONOCIENDO A UN ODROIDIAN - 44

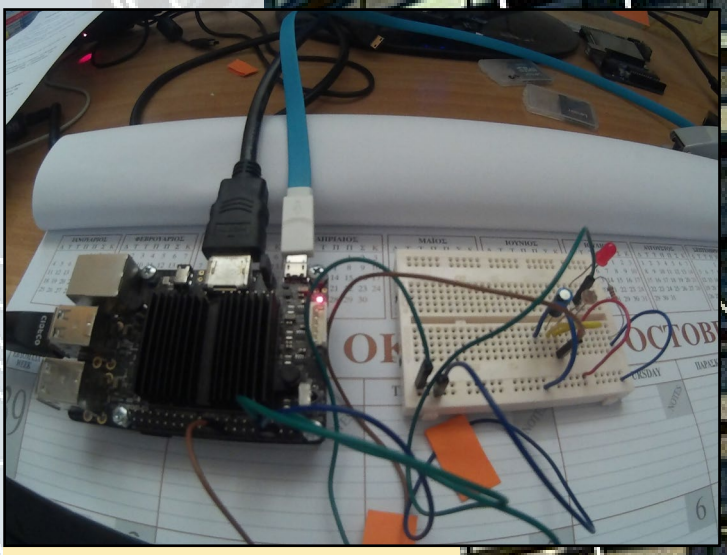
DESARROLLANDO UN DISPOSITIVO IOT CON ODROID-C2

CONTROLADOR DE ILUMINACION DOMESTICA Y ALUMBRADO PUBLICO CON SISTEMA DE NOTIFICACION POR SMS

por Miltiadis Melissas

Todos sabemos que las ciudades consumen mucha energía a la hora de mantener su infraestructura de alumbrado público. Los usuarios normales se enfrentan a una situación similar cuando intentan controlar de un modo eficiente y eficaz la iluminación en sus hogares. La solución de alumbrado IoT que se presenta en este artículo utiliza el ODROIDC2 de Hardkernel, un excelente ordenador de placa reducida (SBC) quad-core de 64 bits (http://bit.ly/2bWxgrK), que puede ayudarte a crear un entorno seguro y eficiente desde el punto de vista energético con funciones inteligentes. Los sensores inteligentes de alumbrado público y de iluminación doméstica pueden conectarse fácilmente a la red con el denominado Internet de las cosas (IoT).

El sensor, una fotorresistencia, puede encender y apagar las luces adecuadamente garantizando así un bajo consumo de energía y un correcto funcionamiento. Además, los usuarios pueden ser avisados desde un dispositivo IoT por medio de mensajes SMS enviados a sus teléfonos móviles. Los mensajes SMS pueden informar a los usuarios del momento exacto en el que se encienden o se apagan las luces de su casa y alertar de posibles malfuncionamientos.



La solución de iluminación integrada utiliza un kit de bricolaje C y ODROID-C2

Este es el tercer proyecto de mi serie de tutoriales relacionados con el Internet de las Cosas (IoT) que utiliza un ODROID-C2. Esta es además, la primera vez que hacemos uso de un sensor de fotorresistencia/fotocélula. Nuestros anteriores proyectos fueron creados y activados usando únicamente accionadores, como LEDs y servomotores. Este artículo te orientará sobre cómo manejar un componente electrónico, controlándolo como una entrada usando la librería WiringPi dentro del lenguaje de programación Python, fijando así las bases para nuestro próximo proyecto IoT: Sistema de notificación y conservación de Vinos.

El dispositivo IoT funciona en condiciones normales de iluminación durante la típica exposición diaria, y la fotorresistencia mantiene el LED apagado en estas circunstancias. Sin embargo, cuando se oscurece, la fotorresistencia activa el ODROID-C2 y el LED se enciende y parpadea, simulando el funcionamiento de las luces de la calle/casa por la noche. Lo interesante es que cuando esto sucede, el ODROID-C2 avisa al usuario que esta operación ha empezado correctamente enviando un mensaje SMS a su teléfono móvil o tablet. Se trata de un completo dispositivo IoT que utiliza un sensor (fotorresistencia), un accionador (LED) y un servicio en la nube (mensajería SMS).

Creando el circuito

Utilizaremos una placa de pruebas para evitar hacer soldaduras y el engorro de tener que diseñar una PCB. Conectaremos varios componentes del circuito con los pines GPIO del ODROID-C2 usando cables conectores Dupont, tal y como muestra la imagen de esta página.

Hardware

ODROID-C2 con Ubuntu
Cable y Fuente de alimentación 5V/2, utiliza el que proporciona la tienda de Hardkernel (http://bit.ly/1XObgdt)
Placa de pruebas y conectores Dupont (macho a hembra)
1 x Fotocélula/Fotorresistencia
1 x Condensador 1uF
1 x LED
1 x Resistencia de 220 ohmios

Software

-Ubuntu 16.04 v2.0 disponible desde Hardkernel en (http://bit.ly/2cBibbk)
-Lenguaje Python para la programación. Afortunadamente Ubuntu 16.04 v2.0 de Hardkernel viene pre-instalado con esta herramienta de programación
-Librería WiringPi para controlar los pines GPIO del ODROID-C2. Para obtener instrucciones sobre cómo instalarlo, consulta la excelente guía de configuración de Hardkernel disponible en `http://bit.ly/2ba6h8o`
- Lenguaje Python para programar el dispositivo IoT

Creando nuestro dispositivo IoT

Como ya hemos mencionado anteriormente, utilizaremos una placa de pruebas para construir nuestro dispositivo IoT con componentes electrónicos y cables conectores Dupont. Se recomienda desconectar la fuente de alimentación del ODROID-C2 antes de conectar cualquier cosa a sus pins, ya que lo puedes dañar con un cortocircuito si realizas una mala conexión accidentalmente. Compruébalo con el esquema de este artículo y realiza las conexiones correctas antes de encenderlo.

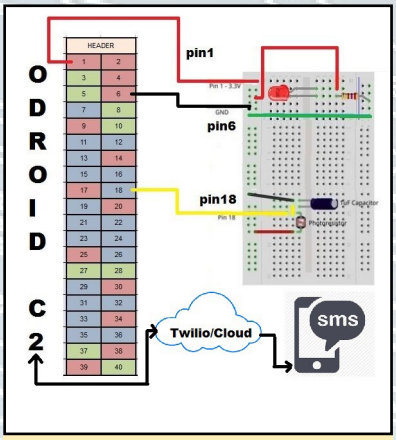


Diagrama del circuito

Para las conexiones usamos cables macho a hembra Dupont. El extremo hembra de este tipo de conectores se conecta a la cabezal macho del ODROID-C2 y el otro extremo macho se conecta a los orificios de la placa de pruebas. Por favor, consulta el esquema de distribución de pines de Hardkernel en la siguiente página para hacer las conexiones, también está disponible en `http://bit.ly/2aXAlmt`:

El Pin físico 1 proporciona la VCC (3,3 V) a nuestro circuito y lo conectamos en la segunda línea vertical de nuestra placa de pruebas.

Puesto que usaremos el Pin 6 como puesta a tierra común, lo conectaremos a la segunda línea vertical de nuestra placa de pruebas, cerca del borde.

La fotorresistencia/fotocélula está conectada al pin físico 18 en uno de sus entremos, el otro va a VCC (3.3V). Ten en cuenta que este cable conector rojo Dupont está conectado a la línea vertical de nuestra placa de pruebas. Vuelve a consultar nuestro esquema para ver cuáles son las conexiones correctas. Debes prestar especial atención a la polaridad del condensador (1uF), ya que necesitamos conectar su extremo negativo marcado con el símbolo (-) a la puesta a tierra. El extremo positivo del condensador se conecta a la fotorresistencia con el cable Dupont amarillo y desde allí al pin físico 18. Explicaremos la función del condensador (1uF) en la siguiente sección.

Finalmente, el LED se coneca al pin físico 7 con su ánodo (+) mientras que el cátodo (-) está conectado por supuesto a la puesta a tierra.

¡Listo! Tenemos todo nuestro cableado físico conectado.

ODROID-C2 40pin Layout

Power Pin
Special Function
GPIO/Special Function

WiringPi GPIO#	Export GPIO#	ODROID-C2 PIN	Label	HEADER		Label	ODROID-C2 PIN	Export GPIO#	WiringPi GPIO#
			3V3	1	2	5V0			
	205	I2CA_SDA	SDA1	3	4	5V0			
	206	I2CA_SCL	SCL1	5	6	GND			
7	249	GPIOX.BIT21	#249	7	8	TXD1	TXD_B	113	
			GND	9	10	RXD1	RXD_B	114	
0	247	GPIOX.BIT19	#247	11	12	#238	GPIOY.BIT10	238	1
2	239	GPIOX.BIT11	#239	13	14	GND			
3	237	GPIOX.BIT9	#237	15	16	#236	GPIOX.BIT8	236	4
			3V3	17	18	#233	GPIOX.BIT5	233	5
12	235	GPIOX.BIT7	#235	19	20	GND			
13	232	GPIOX.BIT4	#232	21	22	#231	GPIOX.BIT3	231	6
14	230	GPIOX.BIT2	#230	23	24	#229	GPIOX.BIT1	229	10
			GND	25	26	#225	GPIOY.BIT14	225	11
	207	I2CB_SDA	SDA2	27	28	SCL2	I2CB_SCL	77	
21	228	GPIOX.BIT0	#228	29	30	GND			
22	219	GPIOY.BIT8	#219	31	32	#224	GPIOY.BIT13	224	26
23	234	GPIOX.BIT6	#234	33	34	GND			
24	214	GPIOY.BIT3	#214	35	36	#218	GPIOY.BIT7	218	27
		ADC.AIN1	AIN1	37	38	1V8	1V8		
			GND	39	40	AIN0	ADC.AIN0		

Distribución de los pines del ODROID-C2

Función de la resistencia y del condensador

Para este circuito, necesitamos 3.3v desde el pin1 del ODROID-C2, además de la Puesta a Tierra (GND), por supuesto. Hemos conectado éstos desde la ODROID-C2 a la placa de pruebas. El LED funcional está conectado al pin7 mediante una resistencia de 220 ohmios para limitar la cantidad de corriente que fluye a través del LED. La presencia de la resistencia asegura que los componentes LED se mantendrán a salvo ante una sobrecarga de corriente accidental.

Sin embargo, el papel del condensador es diferente. Necesitamos que el condensador actúe como un cubo y la fotorresistencia como una fina tubería. Llenar un cubo con un tubo tan delgado lleva mucho tiempo, puedes averiguar su grosor cronometrando el tiempo que lleva llenar el cubo hasta la mitad. En este caso, nuestro cubo es el condensador 1uF. Por lo tanto, la fotorresistencia se conecta a través del condensador de 1µF al pin18 del ODROID-C2 y el polo negativo del condensador se conecta a la toma de tierra. La configuración del hardware ya está terminada, veamos a ver cómo podemos enviar mensajes SMS desde nuestro dispositivo IoT.

Usando Twilio

Twilio es un paquete Python que envía mensajes de texto (SMS). No forma parte de la librería estándar de Python, pero es uno de los miles de paquetes externos de Python que están disponibles para descargar y usar.

Los desarrolladores de Python suelen usar una de las dos utilidades más comunes para descargar y configurar automáticamente las carpetas y archivos necesarios: “easy-install” y “pip”. “Easy-install” viene con la librería setuptools de Python, que es la estándar para Python y “pip” viene con la librería “pip”. “Easy_install” y “pip” se ejecutan en el Terminal y se puede utilizar para instalar paquetes de Python. Puesto que la imagen Linux de ODROID-C2 (http://bit.ly/2cBibbk) viene preinstalada con Python, es muy fácil instalar Twilio en nuestro dispositivo IoT. Lo único que tenemos

que hacer es iniciar la aplicación Terminal y escribir:

```
$ sudo easy_install twilio
```

Introduce la contraseña de administrador para dar permisos a easy-install para que pueda escribir en las carpetas del sistema, que es "odroid" en la imagen oficial Linux de Hardkernel. Por otro lado, si quieres instalar Twilio con pip, que es el instalador para Python, primero tienes que instalar pip en ODROID-C2:

```
$ sudo easy_install pip
$ sudo pip install twilio
```

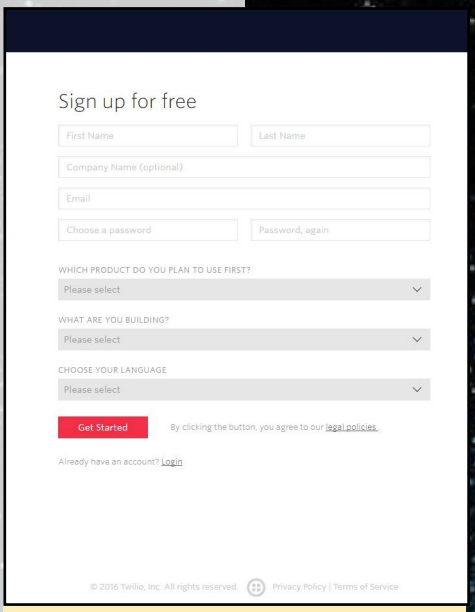
Si quieres comprobar si Twilio se ha instalado correctamente, introduce el comando Python para iniciar el intérprete de Python en el Prompt de comandos o en la aplicación Terminal y escribe estos 2 comandos:

```
> import twilio
> print(twilio._version_)
```

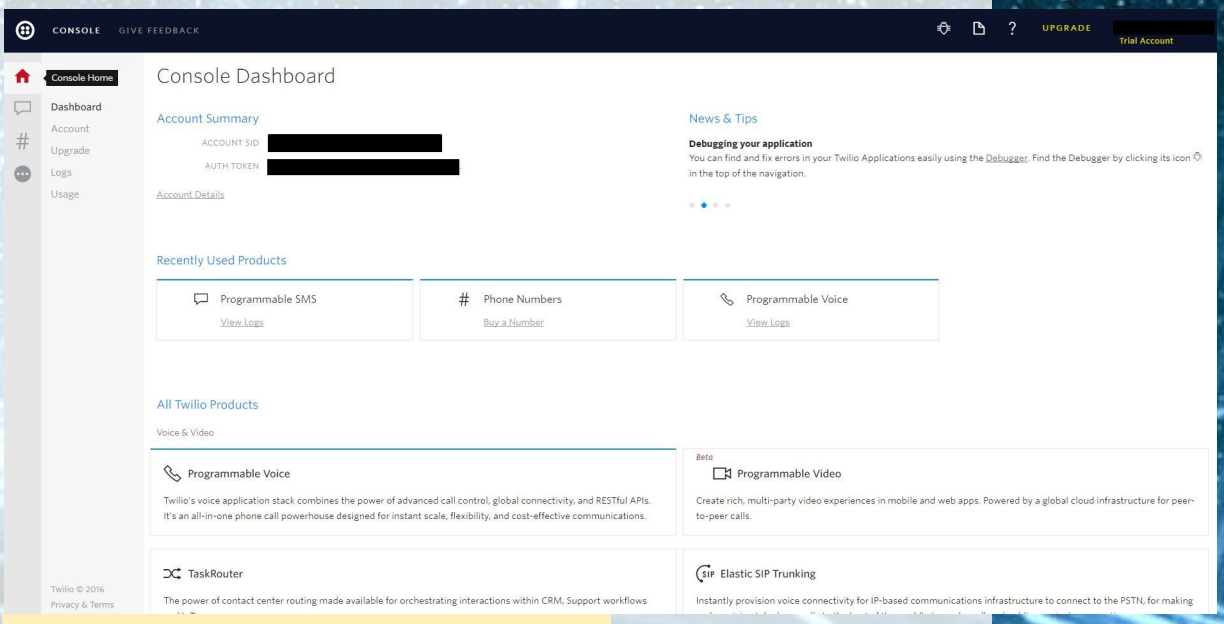
Si aparece un número de versión de Twilio, la instalación se ha realizado correctamente.

Registro en Twilio

Dirígete a la página de registro de Twilio en http://bit.ly/296QVCl y regístrate de forma gratuita, tal y como se muestra en la Figura de la derecha. Twilio necesita tu número de móvil, así que introdúcelo en el campo apropiado. Twilio te enviará un código de verificación al teléfono para verificar que no eres un robot. Introduce el código en la casilla correspondiente y Twilio te proporcionará un número de teléfono. Anota el número de teléfono y continúa con el proceso de registro.



Página de registro de Twilio



Dashboard de Twilio con el token de la API

Finalmente, aterrizarás en una página con muchas opciones, permitiéndote hacer una llamada, enviar un mensaje SMS, recibir una llamada y recibir un mensaje SMS. De esta página, necesitamos el token de autorización de la API de Twilio. Busca el botón “Go to your account “ y haga clic en él. En la página dashboard encontrarás el SID de la cuenta y el token de autorización, tal y como muestra la Figura de la página anterior. A continuación, debes copiarlo y pegarlo en nuestro programa. Ten en cuenta que tu pantalla puede ser algo diferente si es la primera vez que te conectas a Twilio. En mi caso, el token de autorización y el SID de mi cuenta estaban localizados en la parte superior de la página (borrado). En tu caso pueden estar en algún sitio de la parte inferior.

Explicando el código de Twilio

Aqui tienes una muestra de código de la Twilio Python Helper Library disponible en http://bit.ly/2dyGB8n. Todo el código fuente importante para el servicio Twilio está disponible en el repositorio GitHub en `http://bit.ly/2dndZ0s`.

```
from twilio.rest import TwilioRestClient
# Your Account SID from www.twilio.com/console
account_sid = “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”
# Your Auth Token from www.twilio.com/console
auth_token  = “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”

client = TwilioRestClient(account_sid, auth_token)

message = client.messages.create(body=”Hello from ODROID-C2”,
  to=”+306972438526”,    # Replace with your phone number
  from_=”+12052364839”)  # Replace with your Twilio number

print(message.sid)
```

Observarás que dentro de Twilio, hay una carpeta llamada “rest” y dentro de esa carpeta hay una clase llamada “twiliorestclient”. Hacemos uso de esta clase en el siguiente fragmento de código:

```
<from Twilio.rest import Twiliorestclient>
```

En la siguiente línea de código, asignamos una variable client a twilioRestClient para su verificación:

```
<client=twilioRestClient (account_sid, auth_token)>
```

Finalmente, con la siguiente línea, creamos el mensaje y lo imprimimos o lo enviamos a nuestro teléfono móvil:

```
<message=client.sms.messages.create>
```

Enviando el mensaje SMS

Primero, copia y pega el SID de la cuenta y el token de autorización en tu programa. Después, cambia el cuerpo del mensaje de texto a algo como: “Hello from ODROIDC2” como yo lo hice en el siguiente ejemplo. En el campo “to”,

introduce tu número de teléfono. En el campo “from”, tienes que introducir tu número Twilio: este es el número que Twilio te proporcionó al registrarte. Si no lo ha anotado, vuelve a tu cuenta de Twilio (http://bit.ly/2dpPpM1) y en la parte superior de la página encontrarás la pestaña “numbers”, haz clic en ella. Obtendrás tus números de teléfono de Twilio, como se muestra la Figura de esta pagina. Ahora guarda y ejecuta el programa para ver si funciona:

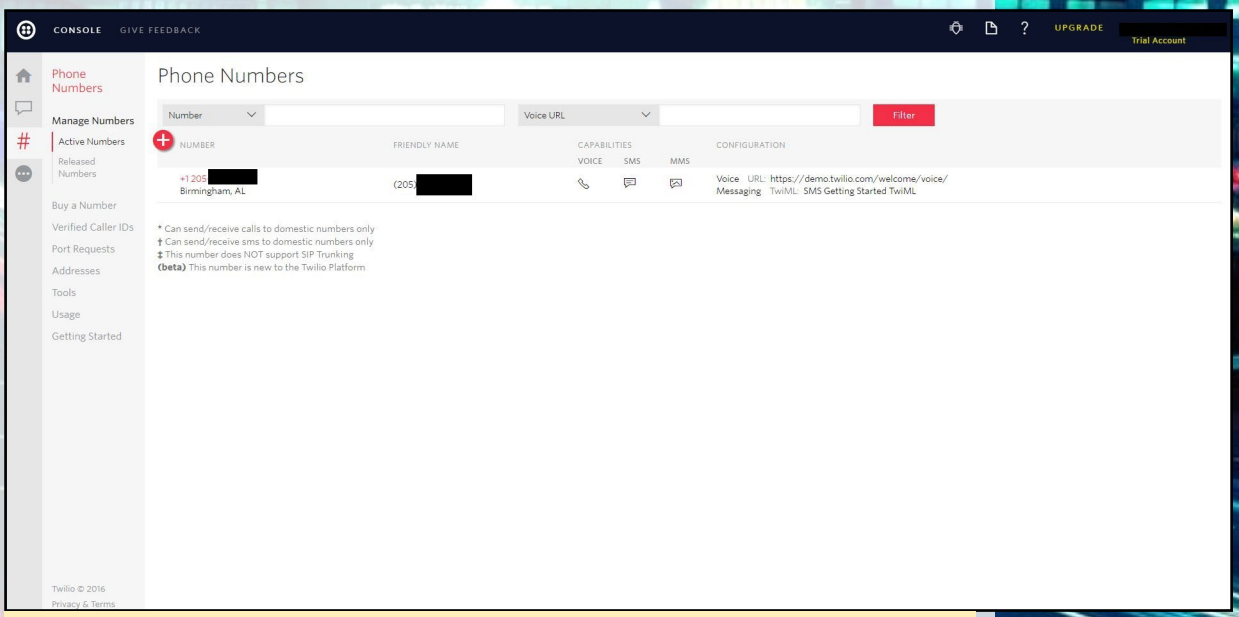
```
from twilio.rest import TwilioRestClient
# Your Account SID from www.twilio.com/console
account_sid = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
# Your Auth Token from www.twilio.com/console
auth_token  = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

client = TwilioRestClient(account_sid, auth_token)

message = client.messages.create(body="Hello from ODROID-C2",
  to="+30XXXXXXXXXX",       # Replace with your phone number
  from_="+120XXXXXXXXX")    # Replace with your Twilio number

print(message.sid)
```

Deberías haber recibido un nuevo mensaje SMS en tu teléfono. El paso final es conectar este código SMS con nuestra fotorresistencia y hacer que nuestro dispositivo IoT sea inteligente.



Pantalla de Twilio que muestra el número de teléfono usado en el fragmento de código

Conectando Twilio a la fotorresistencia

Ahora que sabemos cómo funciona Twilio, vamos a ver cómo conectarlo con el código de la fotorresistencia. La parte difícil es calibrar la fotorresistencia teniendo en cuenta las condiciones de iluminación de la habitación. Siempre hay un umbral que necesitamos descubrir por ensayo y error para activar el dispositivo IoT, el parpadeo del LED y el envío del mensaje SMS al usuario al mismo tiempo. Recuerda que el parpadeo del LED simula el funcionamiento normal de las luces durante la noche, en la calle o en casa, y que el SMS enviado al usuario confirma

este funcionamiento normal. Por favor, analiza el código que se muestra a continuación y sigue las explicaciones línea por línea para ver lo que sucede.

```
#!/usr/bin/env python

# Example for RC timing reading for ODROID-C2
# Must be used with wiringpi2

import wiringpi2 as odroid, time
from twilio.rest import TwilioRestClient

DEBUG = 1
odroid.wiringPiSetup()

LEDpin = 7
odroid.pinMode(LEDpin,1)

def RCtime(RCpin):
        reading = 0
        odroid.pinMode(RCpin,1)
        odroid.digitalWrite(RCpin,0)
        time.sleep(0.1)

        odroid.pinMode(RCpin,0)
        # This takes about 1 millisecond per loop cycle
        while (odroid.digitalRead(RCpin) == 0):
                reading += 1
        return reading

def Send_SMS():
      account_sid = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"  # Your Account
SID from www.twilio.com/console
      auth_token  = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"   # Your Auth Token
from www.twilio.com/console

      client = TwilioRestClient(account_sid, auth_token)

      message = client.messages.create(body="Hello from Python",
              to="+30XXXXXXXXXX",      # Replace with your phone number
              from_="+120XXXXXXXXX")   # Replace with your Twilio number

      print(message.sid)

while True:

     print RCtime(5)     # Read RC timing using physical pin #18
     time.sleep(300)
     if  (RCtime(5)>2500):
       Send_SMS()
     for i in range (0,300):
       if (RCtime(5)>5500):
```

```
            odroid.digitalWrite(LEDpin,1)
            time.sleep(0.02)
            odroid.digitalWrite(LEDpin,0)
            time.sleep(0.02)
```

Primero, importamos el módulo wiringpi2 y creamos el objeto "odroid" porque queremos controlar los pines GPIO de nuestro ODROID-C2:

```
<import wiringpi2 as odroid>
```

Existe un tutorial muy detallado en la excelente página de soporte de Hardkernel sobre cómo descargar e instalar WiringPi en tu ODROID-C2 con Ubuntu en http://bit.ly/2ba6h8o. Despues, importamos TwilioRestClient desde twilio.rest, algo que hemos explicado en detalle en el anterior párrafo.

```
<from twilio.rest import TwilioRestClient>
```

Luego, con la siguiente línea de código, hacemos referencia al cableado GPIO de acuerdo con la tabla proporcionada por Hardkernel para ODROID-C2, tal y como se muestra en la correspondiente figura al inicio de este artículo :

```
<odroid.wiringPiSetup()>
```

Asignamos el pin7 para el LED:

```
<LEDpin=7>
```

Inmediatamente después, lo fijamos como una salida

```
<odroid.pinMode(LEDpin,1)>
```

En la siguiente sección del código, definimos una función llamada RCtime. Esta es una función muy importante para medir los niveles de iluminación en la habitación. Hacemos un seguimiento de estos niveles con un contador:

```
<reading = 0>
```

A continuación, configuramos el pin correspondiente, es decir, el pin5 (pin físico 18) como salida:

```
<odroid.pinMode(RCpin,1)>
```

Luego escribimos hacia ese pin:

```
<odroid.digitalWrite(RCpin,0)>
```

Alternamos el estado casi inmediatamente después usando time.sleep (0.1):

```
<odroid.pinMode(RCpin,0)>
```

El Pin5 queda fijado ahora en "low" y en la siguiente línea de código leemos su estado, añadiendo +1 a nuestro contador:

```
<while (odroid.digitalRead(RCpin) == 0):>
        <reading += 1>
```

Finalmente, comprobamos si el nivel de oscuridad está por debajo del umbral:

```
 <RCtime(5)>2500>
```

El número 2500 se ha calculado realizando varios intentos de ensayo y error buscando el umbral correcto de acuerdo a las condiciones de iluminación de la habitación. Si las condiciones de luz están por encima de este límite, llamamos a la función Send_SMS y enviamos el mensaje SMS a través de Twilio al mismo tiempo que hacemos que parpadee el LED. Ten en cuenta que el LED parpadea durante un intervalo igual al tiempo que fijamos en nuestro dispositivo IoT para comprobar las condiciones de iluminación correctas en la habitación. En este ejemplo, ese intervalo es de 300 segundos, o cada 5 minutos. Por supuesto puedes modificar este intervalo

```
<time.sleep(300)=for i in range (0,300)>
```

Estos intervalos de tiempo deben ser los mismos para garantizar una sincronización correcta. Para que el LED parpadee, el pin se ajusta a high y luego a low con un pequeño intervalo de tiempo de 0.02 milisegundos.

```
for i in range (0,300):
…..
<odroid.digitalWrite(LEDpin,1>
<time.sleep(0.02>
<odroid.digitalWrite(LEDpin,0>
<time.sleep(0.02>
```

Probándolo todo junto

Ahora es el momento de ejecutar y probar nuestro código. Copia y pega el código completo en tu documento IDLE Python (Integrated Development and Learning Environment) con el nombre OdroidSMS.py. Recuerda que todos los scripts de Python tienen la extensión *.py. Puede iniciar IDLE desde tu escritorio Ubuntu simplemente haciendo clic en el menú Aplicaciones (Aplicaciones -> Programación -> IDLE). Una vez que hayas guardado el archivo, ejecútalo con privilegios sudo desde un prompt de comandos después de navegar al directorio donde está ubicado el archivo

```
$ sudo python OdroidSMS.py
```

Esta es la idea básica de un dispositivo inteligente y si yo he podido hacerlo, tú también puedes. ¿Cuál será tu próximo paso? Coge esta guía, analízala cuidadosamente y luego amplíala creando algo más, un dispositivo IoT más sofisticado con tu ODROID-C2.

COMO PODER OBTENER FANTASTICAS IMAGENES DE COLOR DESDE UNA SIMPLE IMAGEN DE SENSOR

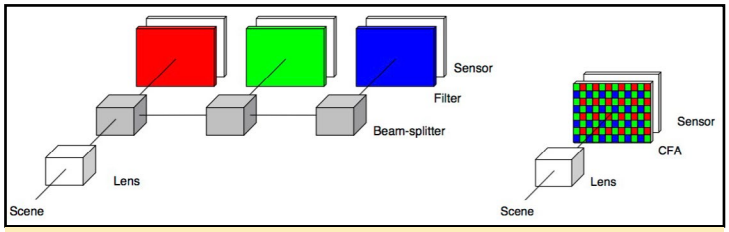
USANDO EL PATRON BAYER PARA CREAR UNA IMAGEN EN COLOR RGB

por withrobot@withrobot.com

Todos sabemos que el color de un píxel se puede representar mezclando tres colores primarios: rojo, verde y azul. Por esto mismo, mucha gente quizás piense que un único píxel del sensor de una cámara también tiene tres colores: rojo, verde y azul. Por ejemplo, en una imagen de 1024 x 1024, generalmente suponemos que hay la misma cantidad de píxeles, 1024 x 1024, de colores rojo, verde y azul. Sin embargo, desde el punto de vista de la fabricación, es muy complicado y costoso poner tres tipos diferentes de sensores de color en una única posición. Por esto, normalmente se utiliza un divisor de haz de luz para iluminar los sensores en diferentes paneles. En consecuencia, esta técnica es muy compleja, voluminosa y costosa.

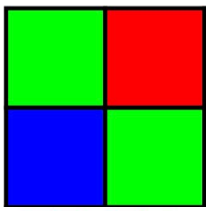
Una alternativa más práctica y asequible es la de disponer de sensores monocromáticos acompañados de un filtro de color. Aquí, el filtro tiene el mismo número de células que los píxeles de la imagen. Por ejemplo, en una imagen de 1024 x 1024, utilizamos 1024 x 1024 sensores monocromos con un filtro de color de 1024 x 1024 celdas con tres colores: rojo, verde y azul. La figura 1 muestra dos diagramas, el que está más a la izquierda es el multi-sensor con divisor de haz de luz. El diagrama de la derecha contiene los sensores monocromáticos con una matriz de filtros de color o CFA.

Aunque se pueden utilizar varios patrones para un CFA, el patrón Bayer es el más común. En la siguiente columna mostramos la forma básica 2x2 del patrón de Bayer que tiene

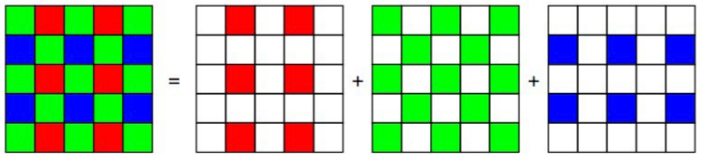


Dos estructuras diferentes del sensor de color: sensores multicolores (izquierda) y sensores monocromos simples (derecha)

dos verdes, uno rojo y otro azul. Utilizamos más subpixeles verdes para simular la sensibilidad del ojo humano que está capacitado para detectar diferentes intensidades del color verde. El patrón Bayer se puede concebir como una combinación de rojo, verde y azul tal y como se muestra a continuación.



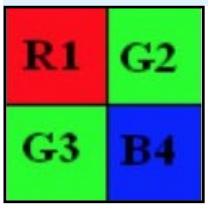
Diseño básico del patrón Bayer: El patrón Bayer es la combinación de los patrones rojo, verde y azul.



Por ejemplo, la primera fila de una imagen 1024x1024 está compuesta de 512 píxeles rojos y 512 píxeles verdes. Del mismo modo, la segunda fila se compone de 512 píxeles verdes y 512 píxeles azules. Por lo tanto, si usamos un byte de datos para cada píxel, el tamaño total de datos de 1 MB de una imagen de 1024 x 1024 está compuesto por 0,25MB de datos de rojo, 0,25MB de azul y 0,5MB de datos de verde. Esto supone una gran reducción del tamaño de los datos de la imagen. El tamaño de los datos de una imagen con una configuración con tres paneles diferentes de sensores de color sería de unos 3 MB o tres veces su tamaño comparado con la imagen de 1 MB del patrón Bayer.

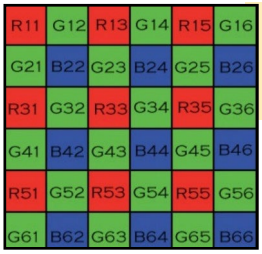
Sin embargo, inevitablemente perdemos información detallada del color usando el sensor de color del patrón Bayer. Por ejemplo, si observamos el píxel superior izquierdo en la figura 3, sólo obtenemos la intensidad del verde. Por consiguiente, tenemos que "adivinar" los otros valores del color para este píxel. Generalmente, se utiliza la interpolación para estimar los valores que faltan. Uno de los métodos más simples es la doble interpolación de píxeles. Utilizando la nomenclatura utilizada en la Figura 4, obtenemos las intensidades de color RGB completas para cada píxel con la siguiente fórmula.

Píxel superior izquierdo: (R, G, B) = (RI, G2, B4)
Píxel superior derecho: (R, G, B) = (RI, G2, B4)
Píxel inferior izquierdo: (R, G, B) = (RI, G3, B4)
Píxel inferior derecho: (R, G, B) = (RI, G3, B4)



Bloque 2x2 del patrón Bayer para la doble interpolación de píxeles.

Aunque obtenemos los datos completos del color con una menor cantidad de cálculo usando esta técnica, también conseguimos una peor calidad de imagen. Para mejorar la calidad de la imagen, se suelen utilizan más píxeles en las proximidades del píxel que se esté completando, además de usar una fórmula más compleja. Un ejemplo de esto es el método de interpolación bilineal.



Bloque 6 x 6 del patrón Bayer para la inerpolación bilinear.

Pixel R33: (R, G, B) =
(R33, (G23+G34+G32+G43)/4, (B22+B24+B42+B44)/4)

Pixel G34: (R, G, B) =
((R33+R35)/2, G34, (B24+B44)/2)

Pixel G43: (R, G, B) =
((R33+R53)/2, G43, (B42+B44)/2)

Pixel B44: (R, G, B) =
((R33+R35+R53+R55)/4, (G34+G43+G45+G54)/4, B44)

Para obtener más información sobre las diferentes técnicas de interpolación, puede consultar "Image Demosaicing: A Systematic Survey" de Xin Li, Bahadir Gunturk y Lei Zhang (http://bit.ly/2eHnGGm).

El problema con la interpolación bilineal es la mala calidad del color. Para superar esta limitación, muchos fabricantes de sensores CMOS utilizan un procesador especial conocido como Image Signal Processor o ISP. Este mejora adicionalmente la imagen obtenida mediante la interpolación del patrón Bayer.

Todos sabemos lo útil que es una cámara con obturador global, como ya quedó patente en el artículo de ODROID Magazine de agosto de 2016, titulado "Conociendo el Obturador Global de la oCam" http://bit.ly/2ee4sJ9, y muchos usuarios de ODROID han solicitado una cámara de obturador global tras el lanzamiento de la oCam-1MGN-U, la cámara monocromática de obturador global. Sin embargo, no se ha contemplado la posibilidad de una cámara de color con obturador global para los ODROIDs porque no existe un sensor de color con un obturador global y la funcionalidad ISP.

Se ha dedicado mucho esfuerzo para solventar este problema usando software, en lugar de esperar a que el hardware apropiado esté disponible. Afortunadamente, se ha desarrollado un nuevo tipo de cámara de color con obturador global para ODROID usando un algoritmo patentado. Esta nueva cámara, la oCam-1CGN-U, estará disponible en diciembre de 2016. La figura 6 muestra la sorprendente mejora en la calidad del color.

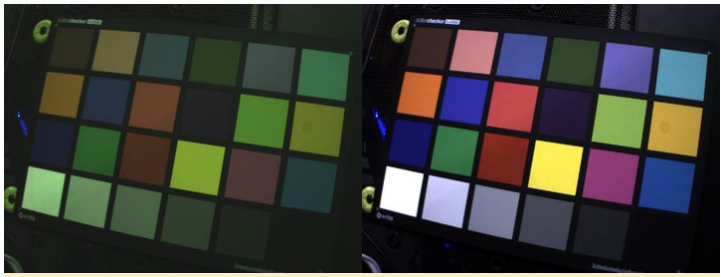
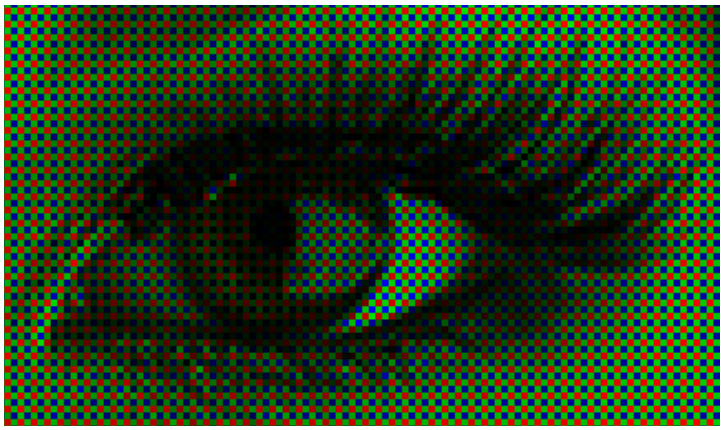


Imagen de color original obtenida usando la interpolación bilineal de la imagen de Bayer (izquierda) y la imagen de color mejorada mediante un algoritmo de mejora patentado (a la derecha) aplicado a la imagen original.

La nueva cámara tendrá las siguientes especificaciones:
Sensor: Sensor de imagen color CMOS OnSemi AR0134 Bayer
Lente: Lente estándar M12 (cambiable)
Tamaño del sensor de imagen: 1/3 pulgadas
Resolución de la imagen: 1280 x 960
Obturador: Obturador global eléctrico
Interfaz: USB 3.0 de super-velocidad

En el artículo del próximo mes, desarrollare un interesante ejemplo utilizando esta nueva cámara con obturador de color global con la plataforma ODROID.



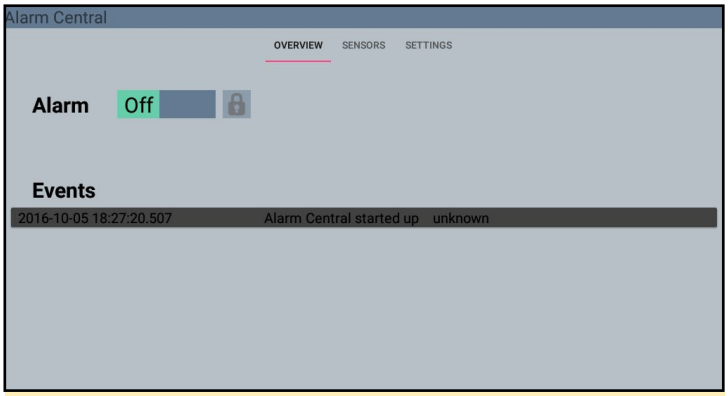
CENTRAL DE ALARMAS

PARTE I - SENSOR DE VENTANA RF24 Y LIBRERIA MIRF

por Jörg Wolff

Esta es la primera parte de una serie de artículos sobre mi proyecto de Central de Alarmas que usa un ODROID-C1 con Android. El proyecto se compone de la aplicación de Android Alarm Central, sensores de ventanas y de movimiento de baja potencia, que aún no están listos. Los sensores se comunican con el ODROID-C1 utilizando los módulos nRF24L01 de 2,4Ghz de Nordic Semiconductor. En este artículo nos vamos a centrar en los sensores de las ventanas de baja potencia y la librería de comunicación mirf, la cual exporte a Android.

Durante mis pruebas iniciales, decidí usar el ODROID-C1 en lugar del ODROID-C2, porque este último no tiene una interfaz SPI nativa y el driver bitbang SPI es demasiado lento para usarlo con los módulos nRF24L01. El desarrollo de las otras partes del sistema: el bloqueo de puertas y el sensor de huella digital, está en marcha. El proyecto Central de Alarmas aún no está instalado en mi casa, pero tan pronto como termine la carcasa para el ODROID-VU7+ y el ODROIDC1, lo instalaré. En caso de activarse una alarma, la aplicación enviará un mensaje corto vía Internet a un teléfono inteligente.



Página de inicio de la central de alarmas

Sensor de ventanas

Los sensores de ventana están basados en un procesador ATTiny84. Diseñé una pequeña placa de 24mm x 60mm que contiene un sencillo enlace eléctrico, un conector para el nRF24L01, un conector ISP para acceder al procesador, un soporte para un CR2450 y algunos componentes adicionales. La PCB fue comprada a Itead Studio y los componentes se soldaron a mano, lo cual me llevo unos 20 o 30 minutos.

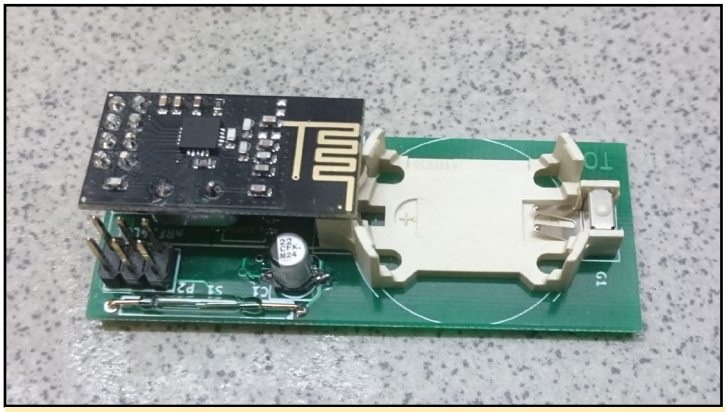
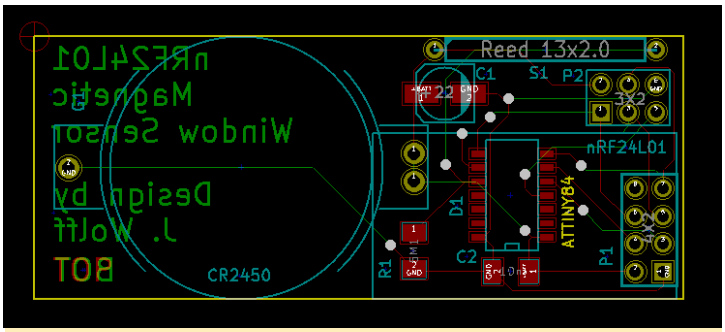


Figura 2 Sensor de ventanas Nrf24



PCB del Sensor de ventanas

Lista de Componentes

- Módulo placa Attiny 84A-SSU SO-14 NRF24L01
- Soporte de batería HU2450 Renata
- Enlace eléctrico NO 13x2.0
- Resistencia 5MI SMD1206
- Condensador 22u/16V 4.3x4.3
- Condensador 10n/50V 3.2x1.6
- Banda de Pin 2x3 2,54
- Tira hembra 2x4 2,54
- Imán Neodym 10x1

(Las dimensiones están en mm)

Sería posible diseñar una placa más pequeña, pero un tamaño mayor permite incorporar una batería CR2450 de 650mAh que proporciona una vida más larga al sistema. El ATtiny está diseñado para entrar en reposo durante unos 4 segundos, luego se reactiva y envía un mensaje de 20 bytes a la Central de Alarmas. Si el enlace eléctrico cambia de estado, el ATtiny se activa y envía un mensaje al ODROID-C1. En modo reposo, el consumo de corriente total de todos los componentes es de aproximadamente 6µA. Cuando el ATtiny se activa, el consumo se eleva durante un corto período a un par de mA. La consumo medio total es de aproximadamente 17 µA. Con una batería de 650mAh esto nos da una vida de unos 3 o 4 años. Sin cifrado de mensajes, el consumo medio sería aún menor y la duración de la batería llegaría a los 5 años. Para alcanzar este consumo tan bajo en modo reposo, el circuito Brown Out Detection está desactivado, lo que hace imposible almacenar datos en la EEPROM. Algunas veces, debido a los reinicios eléctricos, se producen algunas pérdidas de datos como son el número de nodo o la tecla AES. Esto me llevo a tener que implementar el almacenamiento de datos en la memoria flash. Con los datos almacenados en flash, ya no había pérdida de datos durante los reinicios eléctricos.

En el primer arranque de la placa del sensor, envía sus datos sin cifrar con el número de nodo 255. La central de alarmas recibe este mensaje y realiza la numeración automática de nodos y devuelve la clave AES. Esto sólo ocurre cuando la

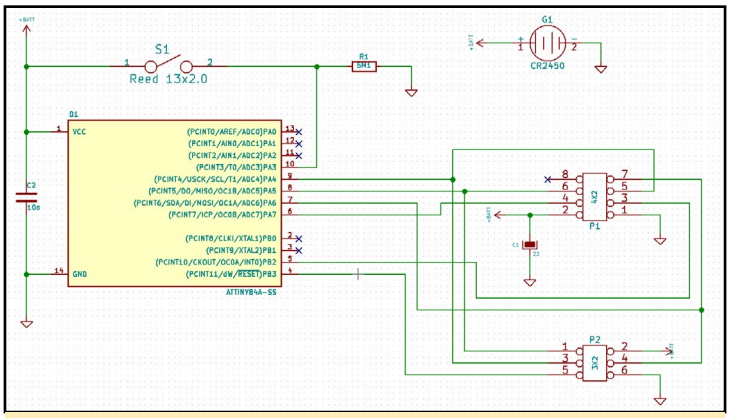
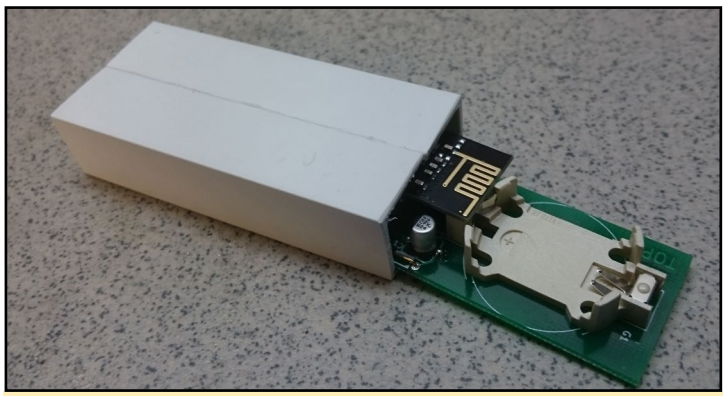


Diagrama del circuito

Central de Alarmas está desconectada y el usuario ha realizado la autentificación. Por un breve periodo de tiempo, la comunicación se abre. El código para el sensor lo puedes encontrar en Github en http://bit.ly/2dHQkrS.

No pude encontrar un pequeño carcasa de plástico que se ajustara a mi sensor. De modo que, utilice un perfil en forma de U de 15mm x 15mm (9/16” x 9/16”) y corté dos piezas de 68 mm (2 5/8”) y las pegué.

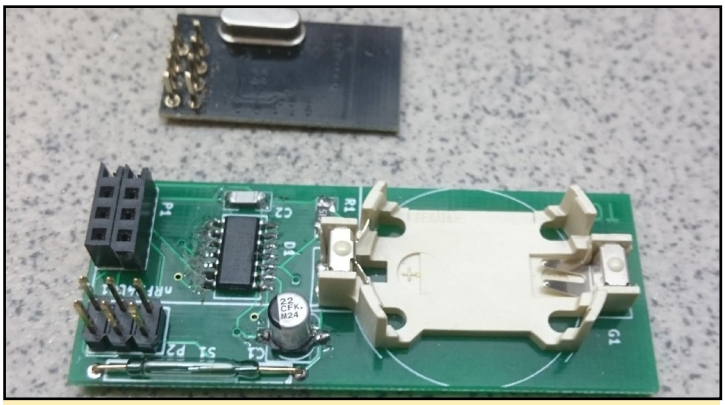
La parte más complicada podría estar en tener que soldar todas los componentes SMD, pero con la técnica correcta y un poco de práctica, no debe haber problema. Lo mejor es comenzar con el ATtiny y sólo soldar un pin para que puedas



Sensor y carcasa

ajustar la posición, luego los otros pines. Funciona muy bien si sueldas algunos pines juntos, ya que una unión de soldadura se puede retirar fácilmente con cordón de desoldadura y un poco de flujo de soldadura. No debes olvidar de limpiar cualquier exceso de flujo de soldadura de los componentes con acetona o un limpiador universal.

Para reducir la altura de la PCB y los componentes, el cuarzo se puede desoldar de la parte superior y soldalo a la parte inferior de la placa nRF24L01. Además, la tira hembra de 2 x 4 se puede humedecer hasta 1 o 2 mm (1/16”) y los pines de la tira de 2 x 4 se pueden cortar unos 2 mm (1/16”). La altura total debe ser de unos 13 mm (½”) para que encaje en la carcasa. El proyecto KiCad lo puedes encontrar en http://bit.ly/2eviBAu.



Sensor y nRF24L01

Librería Mirf

La librería mirf es responsable de la comunicación inalámbrica y ha sido exportada Android. Básicamente, es el mismo código que se utiliza en el ATtiny y el ODROID-C1. Las dos principales diferencias son el código de la interfaz SPI y un contenedor C ++ en el código ODROID. Para compilar la librería de ODROID, primero instala el NDK de Android. A continuación, compila la librería desde la carpeta jni ejecutando el siguiente comando:

```
$ ../../ndk-build -B
```

Puedes encontrar el código fuente en Github en http://bit.ly/2eiANjl. Para usar esta librería en una aplicación de Android, necesitas una librería contenedor como esta:

```
/*
    Model of Mirf wrapping library ported to ODROID-C1 / Android

    Copyright (C) <2016>  <Jörg Wolff>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <jni.h>
#include <stdio.h>
#include <stdlib.h>
#include <android/log.h>
#include <mirf.h>

#ifdef __cplusplus
extern "C" {
#endif

#define LOG_TAG "com.jw.mirf"

#define LOG_D(...) __android_log_print(ANDROID_LOG_DEBUG, LOG_TAG, __VA_ARGS__)
#define LOG_F(fn_name) __android_log_write(ANDROID_LOG_DEBUG, LOG_TAG, "Called : " fn_name )

static JavaVM *java_vm;
mirf* receiver;

jint JNI_OnLoad(JavaVM* vm, void* reserved)
{
    JNIEnv* env;
    if (vm->GetEnv(reinterpret_cast<void**>(&env), JNI_VERSION_1_6) != JNI_OK) {
        return -1;
    }

    // Get jclass with env->FindClass.
    // Register methods with env->RegisterNatives.

    system("insmod /system/lib/modules/spicc.ko");
    system("insmod /system/lib/modules/spidev.ko");

    return JNI_VERSION_1_6;
}

//mirf(uint8_t _cepin, uint32_t _freq, uint8_t _spi_channel, uint8_t _payload_size, uint8_t _mirf_CH);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfSetup(JNIEnv * env, jobject obj, uint8_t ce, uint32_t speed, uint8_t spi_channel, uint8_t size, uint8_t mirf_channel) {
    receiver = new mirf(ce, speed, spi_channel, size, mirf_channel);
    //LOG_D("Setup");
}

//void config(void);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfConfig(JNIEnv* env, jobject obj) {
    if (receiver != NULL) receiver->config();
}

//void reconfig_rx(void);
JNIEXPORT void JNICALL
```

```
Java_path_to_your_app_MirfReConfigRx(JNIEnv* env, job-
ject obj) {
    if (receiver != NULL) receiver->reconfig_rx();
}


//void reconfig_tx(void);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfReConfigTx(JNIEnv* env, job-
ject obj) {
    if (receiver != NULL) receiver->reconfig_tx();
}


//void set_address(uint8_t pos, uint8_t* address);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfSetAddress(JNIEnv* env,
jobject obj, jbyte pos, jstring address) {
    if (receiver != NULL){
        const char *nativeString = env-
>GetStringUTFChars(address, 0);
        receiver->set_address(pos, (uint8_t*)na-
tiveString);
        LOG_D("SetAddress: %s", nativeString);
        env->ReleaseStringUTFChars(address, na-
tiveString);
    }
}


//uint8_t receive_data(void* buf);
JNIEXPORT jbyteArray JNICALL
Java_path_to_your_app_MirfReceiveData(JNIEnv* env,
jobject obj, jbyte size) {
    if (receiver != NULL) {
        jbyte *data=(jbyte *)
malloc(size*sizeof(jbyte));
        receiver->receive_data(data);
        jbyteArray result=env->NewByteArray(size);
        env->SetByteArrayRegion(result, 0, size,
data);
        delete[] data;
        return result;
    }
    return 0;
}


//uint8_t transmit_data(void* buf);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfTransmitData(JNIEnv* env,
jobject obj, jbyteArray array) {
    if (receiver != NULL) {
        jbyte *buf = env->GetByteArrayElements(array,
NULL);
        receiver->transmit_data(buf);
        env->ReleaseByteArrayElements(array, buf, 0);
    }
}


//uint8_t status(void);
//uint8_t max_rt_reached(void);


//uint8_t data_ready(void);
JNIEXPORT int JNICALL
Java_path_to_your_app_MirfDataReady(JNIEnv* env, job-
ject obj) {
    if (receiver != NULL) return receiver->data_
ready();
    LOG_D("MirfDataReady:return 0");
    return 0;
}


//uint8_t read_register(uint8_t reg, uint8_t* buf,
uint8_t len);
//uint8_t read_register(uint8_t reg);
//uint8_t write_register(uint8_t reg, const uint8_t*
buf, uint8_t len);
//uint8_t write_register(uint8_t reg, uint8_t value);
//void config_register(uint8_t reg, uint8_t value);
//uint8_t get_data(void* buf);
//uint8_t send_data(void* buf);


//void power_up_rx(void);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfPowerUpRx(JNIEnv* env, job-
ject obj) {
    if (receiver != NULL) receiver->power_up_rx();
}


//void power_up_tx(void);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfPowerUpTx(JNIEnv* env, job-
ject obj) {
    if (receiver != NULL) receiver->power_up_tx();
}
```

```
//void power_down(void);


//uint8_t flush_rx(void);
JNIEXPORT int JNICALL
Java_path_to_your_app_MirfFlushRx(JNIEnv* env, job-
ject obj) {
    if (receiver != NULL) return receiver->flush_rx();
    return 0;
}


//uint8_t flush_tx(void);
JNIEXPORT int JNICALL
Java_path_to_your_app_MirfFlushTx(JNIEnv* env, job-
ject obj) {
    if (receiver != NULL) return receiver->flush_tx();
    return 0;
}


//void start_listening(void);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfStartListening(JNIEnv* env,
jobject obj) {
    if (receiver != NULL) receiver->start_listen-
ing();
}


//void stop_listening(void);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfStopListening(JNIEnv* env,
jobject obj) {
    if (receiver != NULL) receiver->stop_listening();
}


//void delay_us(unsigned int howLong);
JNIEXPORT void JNICALL
Java_path_to_your_app_MirfDelayMicroSeconds(JNIEnv*
env, jobject obj, int us) {
    if (receiver != NULL) receiver->delay_us(us);
}


#ifdef __cplusplus
}
#endif
```

La librería Mirf y las funciones necesitan algo de código glue, así que la aplicación Java puede llamar a la librería C++:

```
static {
    System.loadLibrary("mirf_android");
}


public native int MirfSetup( byte ce, int speed, byte
spi_channel, byte size, byte mirf_channel);
public native void MirfConfig();
public native void MirfReConfigTx();
public native void MirfReConfigRx();
public native void MirfPowerUpRx();
public native void MirfPowerUpTx();
public native void MirfSetAddress(byte pos, String
address);
public native byte[] MirfReceiveData(int size);
public native void MirfTransmitData(byte[] data);
public native int MirfDataReady();
public native void MirfStartListening();
public native void MirfStopListening();
public native int MirfFlushRx();
public native void MirfDelayMicroSeconds(int us);
```

Y para crear un objeto mirf, este código como ejemplo en la función onCreate ():

```
/*
* Some needed constants for the mirf object
*/
byte pin_ce = 6; //Header pin 22
byte spi_channel = 0;
byte length_payload = 20;
byte mirf_channel = 5;
int spi_speed = 4000000;
/*
* Setup the mirf communication
*/
MirfSetup(pin_ce, spi_speed, spi_channel, length_pay-
load, mirf_channel);
MirfConfig();
```

En un bucle o un HandlerThread, los mensajes se pueden leer desde los sensores, como se muestra en el siguiente fragmento de código:

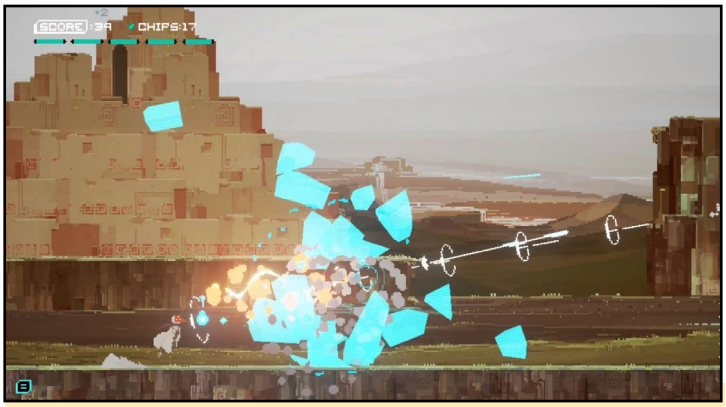
ANCESTOR

UN JUEGO REPLETO DE DIVERSION CON UNA JUGABILIDAD, DETALLES Y EFECTOS VISUALES PERFECTOS

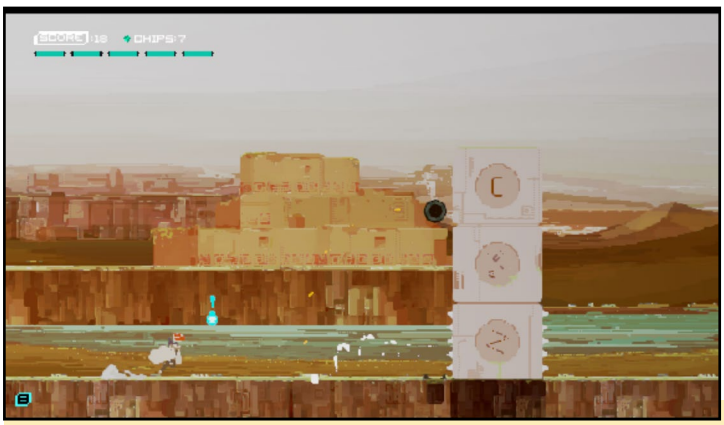
por Bruno Doiche

Mientras que nosotros y nuestros queridos ODROIDs aún estamos muy lejos de poder emular la Playstation 3, donde podemos disfrutar jugando a Journey hasta el agotamiento, podemos conseguir un juego con similares efectos visuales y rompecabezas, ¡y la ventaja añadida de ser un juego repleto de líderes! Ancestor fue escrito por un equipo de producción que está ligado al muy querido Mass Effect. Se asociaron con su padre, un programador y el resto depende de ti para averiguar hasta dónde puede llegar con este juego tan absolutamente divertido ¡Coge tu joystick y piensa rápido!

https://play.google.com/store/apps/details?id=com.supermegaquest.ancestor



El aspecto es similar a Journey, aunque la jugabilidad es frenética



En Ancestor, resuelves rompecabezas para progresar y divertirte


```
while (true) {
    MirfPowerUpRx();
    MirfFlushRx();
    MirfStartListening();
    while (MirfDataReady() == 0) {
        MirfDelayMicroSeconds(250);
    }
    MirfStopListening();


    inbuffer = Arrays.copyOf(MirfReceiveData(length_
payload), 16);


    //Do something with inbuffer.


    try {
        Thread.sleep(5L);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

El cableado del módulo nRF24L01 al ODROIDC1 es el siguiente:

CI.Header.19 – nRF24L01.6	**(MOSI)**
CI.Header.21 – nRF24L01.7	**(MISO)**
CI.Header.23 – nRF24L01.5	**(SCK)**
CI.Header.22 – nRF24L01.3	**(CE)**
CI.Header.24 – nRF24L01.4	**(CSN)**
CI.Header.1 – nRF24L01.2	**(VCC)**
CI.Header.6 – nRF24L01.1	**(GND)**

En la siguiente parte de esta serie, compartiré más información sobre el sensor de movimiento RF24, la propia aplicación Central de alarma y un curiosa carcasa hecho a mano para el ODROID-VU7 +..

AMBILIGHT ULTRA-HD 4K

CREA UN ESPECTACULAR TRASFONDO VISUAL SINCRONIZADO CON TU SISTEMA DE CINE EN CASA

por Charles Park y Brian Kim

Ambilight, abreviatura de "ambient lighting", es un sistema de iluminación para televisores desarrollado por Philips que permite crear efectos de iluminación alrededor del televisor sincronizados con el contenido del vídeo. Puedes conseguir un efecto similar con ayuda de una banda de LEDs RGB y un software que muestre la imagen en pantalla, luego asigna un color a cada LED individual con diferentes tonalidades según corresponda. En este artículo, vamos a ver cómo usar un ODROID para lograr esto.

Figura 1 - Ambilight en el ODROID-C2

Requisitos

Utilizamos un Arduino para controlar los LEDs y un ODROIDC2 para ejecutar el reproductor multimedia Kodi como interfaz para el reproducir contenido multimedia. Los componentes de hardware que necesitamos son:

- ODROID-C2
- Arduino UNO
- Cable USB Tipo A - Tipo B
- Fuente de alimentación 5V/6A
- Módulo eMMC de 32 GB para el C2 con Linux
- LEDs WS2801
- Cable DC con clavija de 2.5mm
- Cable conector de 4 pines x 3

Usaremos un Arduino para controlar los LEDs y ODROID-C2 para hacer todo lo demás, como es reproducir el archivo multimedia, capturar la imagen de video y enviar los datos de los colores LED a través de la interfaz serie USB.

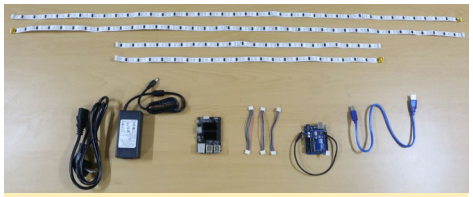


Figura 2 - Componentes de hardware para Ambilight

Arduino recibe los datos de los colores desde ODROID-C2 y luego ajusta el color en cada LED.

Configuración de hardware

El primer paso es configurar el cableado en la placa Arduino. Existen dos zonas de cableado: el conector LED de 4 pines y el cable DC con clavija del ODROID-C2. El conector LED de 4 pines se conecta a los LEDs WS2801 que se pueden controlar a través de la interfaz SPI.

Cable conector LED de 4 pines

- Rojo: VCC
- Negro: Puesta a Tierra
- Azul: SCK (13)
- Verde: MOSI (11)

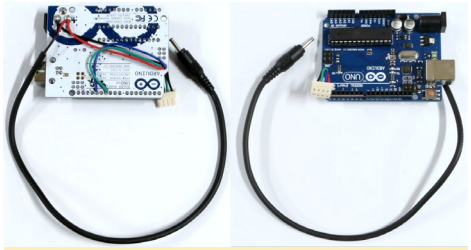


Figura 3 - Cableado Arduino

Cable DC con clavija

- Rojo: VCC
- Negro: Tierra

El WS2801 es un controlador LED en constante desarrollo, que está di-señado para pantallas LED interiores y exteriores y los sistemas de iluminación LED decorativos. Para montar los LEDs en el televisor, cortaremos las tiras de LEDs con un tamaño algo menor a la altura y anchura del televisor. Cada rollo de LED se debe conectar con cables conectores de 4 pines.

Configuración del software

Son tres los tipos de software que necesitamos para montar nuestro ambilight casero: el firmware que controla el Arduino, Hyperion y el reproduc-

Figura 4 - Cableado del LED

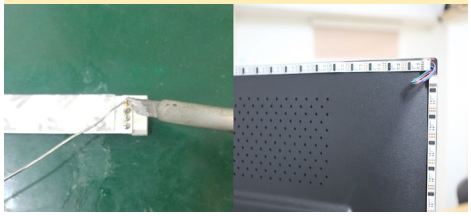


Figura 5 - Ambilight montado en el TV

tor multimedia Kodi. El Arduino está conectado al ODROID-C2 a través del dispositivo serie USB (ttyACM0). Podemos desarrollar con facilidad el firmware de Arduino en Linux de forma nativa sobre el ODROID-C2 usando el IDE de Arduino:

```
$ sudo apt-get update
$ sudo apt-get install arduino
$ cd /usr/share/arduino/librar-
ies/
$ sudo git clone \
  https://github.com/adafruit/
Adafruit_NeoPixel.git
```

La librería Adafruit NeoPixel permite controlar los LEDs. El código fuente del firmware para controlar los LEDs se puede descargar de https://git.io/vPVqT:

```
$ wget https://git.io/vPVqT -O
odlight.ino
$ arduino
(Ctrl + O) -> (Select /home/
odroid/odlight.ino file)
(Ctrl + R) Verify / Compile
(Ctrl + U) Upload
```

Por supuesto, Arduino necesita estar conectado al ODROID-C2 durante la carga del firmware. Podemos obtener más información sobre el software Arduino IDE en la página principal de Arduino en http://bit.ly/212hc7p.

Para describir el comportamiento de Ambilight, el software de fondo captura la imagen de vídeo al mismo tiempo que se reproduce el archivo multimedia, a continuación, envía los datos RGB al dispositivo que controla los LEDs. Hyperion es una implementación de código abierto de Ambilight que se ejecuta en muchas plataformas y que utiliza el controlador de captura de vídeo del ODROID-C2 para obtener los datos de la imagen de vídeo. Sin embargo, el controlador que captura el vídeo asigna un área DMA de 8 megabytes, de modo que el kernel linux de ODROID-C2 necesita más memoria contigua:

```
$ sudo apt-get update
$ sudo apt-get install git build-
essential
$ git clone --depth 1 \
  https://github.com/hardkernel/
linux.git -b odroidc2-3.14.y
$ cd linux
$ make odroidc2_defconfig
$ make menuconfig
Device Drivers --->
  Amlogic Device Drivers --->
    Video Decoders --->
      [*] Amlogic Video Capture
support
  Generic Driver Options --->
    *** Default contiguous memory
area size: ***
      (12) Size in Mega Bytes
$ make -j4
$ sudo make modules_install
$ sudo mv /media/boot/Image /
media/boot/Image.back
$ sudo mv /media/boot/ /media/
boot/Image.back
$ sudo mv /media/boot/meson64_
odroidc2.dtb \
  /media/boot/meson64_odroidc2.
dtb.back
$ sudo cp arch/arm64/boot/Image /
media/boot/
$ sudo cp arch/arm64/boot/dts/me-
son64_odroidc2.dtb \
  /media/boot/
$ sudo sync
$ sudo reboot
```

Hyperion es una buena opción como software para controlar el color de los LEDs, ya que requiere poca potencia de procesamiento, su funcionamiento es rápido y eficaz, y su configuración es sencilla. Además, Hyperion es compatible con la plataforma Amlogic en el ODROID-C2, aunque todavía no lo soporta oficialmente. No obstante, no es complicado añadir soporte para el ODROID-C2 en Hyperion:

```
$ sudo apt-get update
$ sudo apt-get install cmake
libqt4-dev \
  libusb-1.0-0-dev python-dev
libxrender-dev \
  python libasound2-dev zlib1g-
dev
$ git clone --depth 1 \
  https://github.com/mdrjr/c2_
aml_libs.git
$ cd c2_aml_libs
$ sudo make
$ sudo make install
$ cd
$ git clone --depth 1 \
  --recursive https://github.com/
bkrepo/hyperion.git
$ cd hyperion
$ mkdir build
$ cd build
$ cmake -DENABLE_DISPMANX=OFF \
  -DENABLE_SPIDEV=OFF -DENABLE_
AMLOGIC=ON \
  -DCMAKE_BUILD_TYPE=Release
-Wno-dev ..
$ make -j4
$ sudo make install
```

Hyperion necesita un archivo de configuración, que se puede general fácilmente con el programa de configuración de Hypercon, que está disponible en http://bit.ly/2dRqkgO. Incluso si Hypercon no puede generar un

Figura 6 - Hypercon

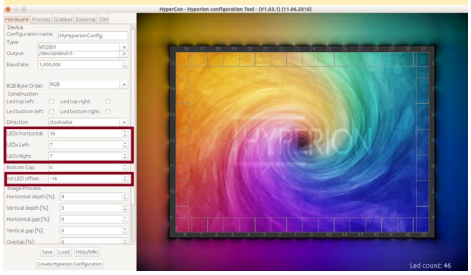


Figura 7 – Ambilight en funcionamiento

archivo de configuración completo para ODROID-C2, el programa es útil para configurar la posición de los LEDs. Existen tres opciones para configurar las posiciones de los LED: LEDs horizontales, LEDs a la izquierda y LEDs a la derecha. La primera opción de LED es para ajustar el punto de inicio del LED, también podemos configurar la dirección en sentido a las agujas del reloj o al contrario. Para obtener el archivo de configuración en formato JSON una vez finalizada la configuración de la posición de los LEDs, simplemente haz clic en el botón Create Hyperion Configuration.

La opción "leds" en el archivo de configuración JSON generado a través de Hypercon se tiene que copiar al archivo de configuración por defecto (https://git.io/vPrKR). El resto de opciones del archivo de configuración JSON pueden permanecer con los valores por defecto:

```
$ wget https://git.io/vPovU -O /
etc/hyperion/hyperion.config.json
```

Abre el archivo de configuración generado con Hypercon, después sobreescribe la opción "leds" desde archivo de configuración generado y escríbelo en el archivo /etc/hyperion/hyperion.config.json.

Reproducir un video

Para ejecutar Ambilight en el ODROIDC2, ejecuta el demonio Hyperion en segundo plano, luego reproduzce una película en Kodi:

```
$ hyperiond /etc/hyperion/hyper-
ion.config.json &
$ kodi
```

El ODROID-C2 también soporta películas en 4K H265, lo cual requiere configurar la opción double_write_mode, tal y como se detalla en http://bit.ly/2doJtDU:

```
$ echo 1 | sudo tee \
  /sys/module/amvdec_h265/param-
eters/double_write_mode
```

Para obtener más información, consulta el proyecto Adalight http://bit.ly/1EnG6zZ), nuestro artículo anterior de Ambilight en ODROID Magazine (http: // bit.ly/2dOPWsk) y nuestros hilos del foro ODROID en http://bit.ly/2eyPrUr and http://bit.ly/2eo1DrY.

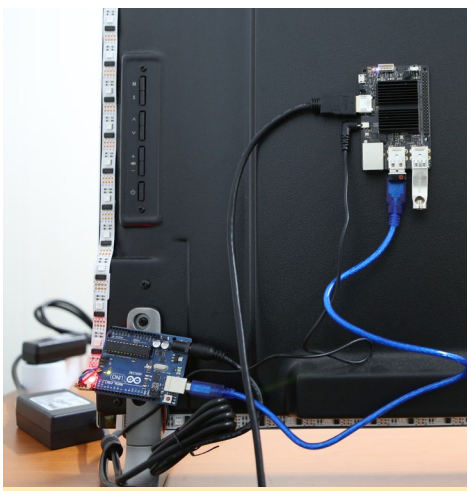


Figura 8 - Primer plano de Ambilight

DOCKER 101

PARTE I - ¿POR QUE DOCKER?

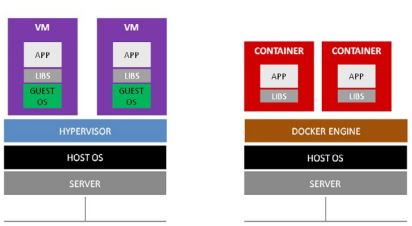
por Andy Yuen

El artículo "Why People user Docker" (http://bit.ly/2f5nRyi) nos habla de Docker largo y tendido. Me interesa Docker porque me permite montar y duplicar entornos aislados de un modo muy simple. Puedo crear un entorno en tiempo de ejecución una vez, empaquetarlo y luego ejecutarlo de nuevo en cualquier otra máquina. Además, todo lo que se ejecuta en ese entorno está aislado del host subyacente (igual que una máquina virtual). Y lo mejor, es un proceso rápido y sencillo.

Docker utiliza una tecnología de virtualización llamada contenedores. Los contenedores usan la función del kernel Linux "cgroups" para aislar contenedores y otros procesos que se ejecutan en el host y "namespaces" para hacer accesible un conjunto de recursos del sistema y presentarlos a un proceso como si estuvieran dedicados exclusivamente a ese proceso. La tecnología de contenedor es diferente de la tecnología de máquina virtual en cuanto que no necesita un hipervisor ni tampoco un sistema operativo invitado. Los contenedores sólo requieren una aplicación empaquetada con librerías y binarios dependientes en una imagen. Para las máquinas virtuales, la virtualización se hace a nivel del dispositivo, mientras que para los contenedores ésta sólo desciende hasta el nivel del sistema operativo.

Figura 1 - Máquina virtual vs contenedor

Virtual Machine vs Container



Comparados con las máquinas virtuales, los contenedores son más pequeños y consumen menos recursos mientras que su rendimiento está mejorado. Puesto que tu aplicación está aislada en una imagen Docker junto con sus dependencias (binarios y librerias), puedes compilarla una vez y ejecutarla en cualquier host Docker con la misma arquitectura del equipo en el que se creó la imagen. Esto significa que las imágenes de Docker desarrolladas para la arquitectura Intel x86 no se pueden ejecutar en máquinas ARM64 y viceversa.

Existen muchas más razones por las que las empresas usan Docker CI/CD, DevOps, implementación azul/verde, actualizaciones sucesivas, etc. Simplemente no tengo espacio en este artículo para detallarlas todas. Un dato muy interesante sobre Docker es que cada vez que usas Google para realizar un búsqueda, consultar Gmail o Google Docs, se despacha un nuevo contenedor.

Usando Docker

La Figura 2 muestra el flujo de trabajo con Docker. La siguiente lista detalla el flujo:

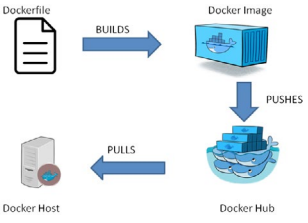
1. Alguien crea un Dockerfile
2. El Dockerfile se utiliza para crear una imagen, que es tu aplicación con sus binarios y librerías dependientes.
3. La imagen se envía a Docker Hub, que actúa como un repositorio central para imágenes Docker. Puedes encontrar un gran número de imágenes Docker que puede descargar y utilizar si se adaptan a tus necesidades.
4. Un usuario extrae una imagen de Docker Hub y ejecuta la imagen en un contenedor. Se pueden ejecutar multiples copias de la aplicación en el mismo host de Docker o en un clúster de Docker según se necesite.

Información general

En este tutorial, trataremos todas las actividades que hemos indicado anteriormente y que hemos dividido en 2 partes. La primera parte estará dedicada a los clásicos comandos de Docker para iniciar y detener los contenedores y permitir que se comuniquen entre sí. La segunda parte cubre el modo swarm de Docker, que es nuevo en la versión 1.12 de Docker. El modo Swarm se centra en la planificación, es decir en la agrupación

Figura 2 - Usando Docker

Using Docker



y la programación de dónde se ejecutan los contenedores y cuántas réplicas de éstos deberían iniciarse en un clúster.

A principios de 2015 se publicaron varios artículos en ODROID Magazine sobre docker, ¿qué ha cambiado desde entonces? Por un lado, los kernels que soportaban Docker no eran tan comunes en los típicos sistemas operativos para ODROID. Para ejecutar Docker, se tienen que hacer pequeños ajustes en el kernel que, para la mayoría de la gente, incluido yo, representan una molestia. Sin embargo, las funciones del kernel que son necesarias para Docker se han incluido en muchos sistemas operativos. Otra cosa a tener en cuenta es que Docker ha sido diseñado y desarrollado para máquinas de 64 bits. En 2015, todos los ODROIDs estaban basados únicamente en la arquitectura de 32 bits. El ODROID C2 es el primer ODROID de 64 bits hasta la fecha. Aunque Docker puede ser y ha sido, adaptado para ejecutarse en arquitectura ARM de 32 bits, esta es la primera implementación en 64 bits. Vamos a explorar sus posibilidades en este tutorial.

Requisitos previos

Para continuar con este tutorial, debes contar con lo siguiente:

1. ODROID-C2 ejecutando un sistema operativo con las características del kernel necesarias para activar Docker

Yo he utilizado el servidor Xenial de Armbian, basado en Ubuntu. Cómo decidí usar este servidor, deje documentado el proceso en mi blog en http://bit.ly/2dyTUGr. Recientemente he echado un vistazo al sitio web de Armbian en http://bit.ly/2exEWPH y me he dado cuenta que únicamente el servidor Jessie, basado en Debian, es el que está disponible para descargar.

2. ODROID-C2 con motor Docker instalado.

Para instalar el motor Docker en tu máquina, usa los siguientes comandos:

```
$ docker run -d -p 3306:3306 \
  --name mysql \
  -e MYSQL_USER=fishuser \
  -e MYSQL_PASSWORD=fish456 \
  -e MYSQL_DATABASE=fish \
  -v /media/sata/fish-mysql:/u01/
my3306/data \
  mrdreambot/arm64-mysql
```

Si estás utilizando un sistema operativo diferente al mío, tiene que usar el comando de gestión de paquetes de tu sistema operativo: yum, dnf o apt-get. Si Docker no está disponible en el repositorio de tu sistema operativo, prueba los binarios Docker v1.12.1 que yo utilicé junto con el script de instalación de http://bit.ly/2ejY1FE. Estos binarios han sido probados en los servidores Xenial y Jessie de Armbian. Para la Parte 1 del tutorial, Docker.io v1.10, v1.11 o v1.12 funcionarán bien. Para la Parte 2 necesitarás la v1.12.

Para evitar tener que añadir "sudo" delante de cada comando de Docker que utilices, añade tu nombre de usuario (login) al grupo "docker", luego reinicia el sistema y vuelva a iniciar sesión:

```
$ docker run -d \
  -p 8080:8080 --name fish \
  -e MYSQL_SERVER=192.168.1.100 \
  -e MYSQL_PORT=3306 \
  mrdreambot/arm64-fish
```

3. ODROID-C2 con conexión a Internet

Tu C2 necesitará una conexión a Internet para bajar las imágenes desde Docker Hub durante el tutorial

Comandos básicos de Docker

En el tutorial, para los comandos simples cuyo resultado sea autoexplicativo, sólo incluiré las capturas de pantalla para el comando y el resultado. Para los comandos más complejos, también explicaré la sintaxis del comando y el significado de las opciones. Si necesitas ayuda con las opciones disponibles de un determinado comando, simplemente escriba el comando y añade --help como se muestra en la Figura 6.

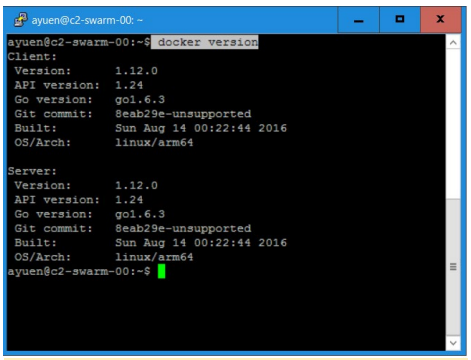


Figura 3 - Comprobando la versión de Docker

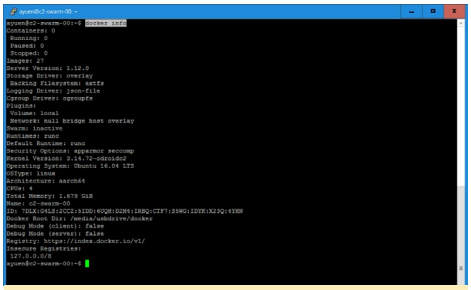


Figura 4 – Buscando más info sobre Docker

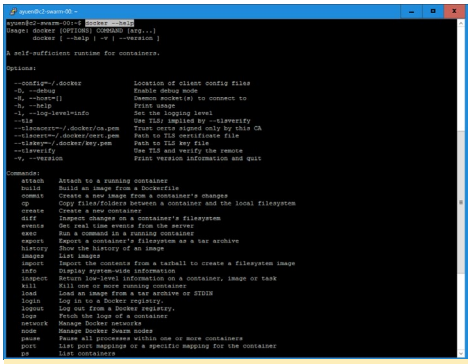


Figura 5 - Obteniendo ayuda

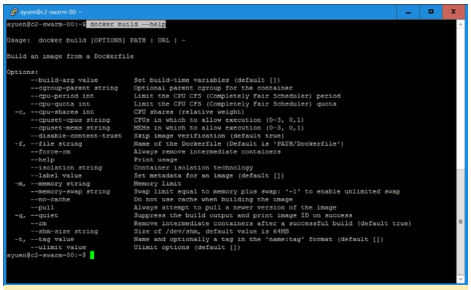


Figura 6 - Ayuda de Docker

Ejecutando tu primer contenedor

Asegúrate de que tiene una conexión de Internet e introduce el siguiente comando:

```
$ docker run -d -p 80:80 --name
```

```
httpd \
  mrdreambot/arm64-busybox-httpd
```

Opciones:

-d significa que se ejecuta en modo demonio

-p 80:80 significa el puerto 80 del contenedor hacia el puerto 80 del host, de modo que puedes tener acceso a la aplicación en el puerto 80 de tu C2.

--name proporciona un nombre al contenedor que acabas de iniciar. Esto es opcional, aunque te recomiendo que siempre le des un nombre para que puedas referirte a tu contenedor con facilidad. Puede identificar los contenedores que están ejecutándose incluso sin utilizar la opción --name usando el comando "docker ps", que se describe más adelante. Puesto que la imagen "mrdreambot/arm64-busybox-httpd" no está en tu host Docker, Docker la descargará desde Docker Hub.

Navega con tu navegador de tu PC a tu host Docker, que es tu ODROID-C2. Deberías ver la página que se muestra en

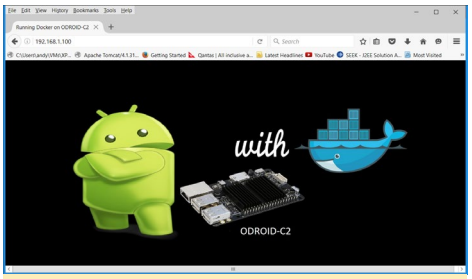


Figura 7 - Página de prueba Docker

la Figura 7, que es nuestro programa Docker "Hello World".

Conectando a tu contenedor activo

Puedes conectarte a tu contenedor en funcionamiento con el siguiente comando:

```
$ docker exec -it httpd /bin/sh
```

Las opciones son las siguientes:

-it significa modo tty interactivo

httpd es el nombre que le dimos al contenedor cuando lo iniciamos

/bin/sh es el comando que queremos ejecutar cuando nos conectamos al contenedor. En este caso, queremos iniciar el interprete de comandos.

Podemos ejecutar cualquier comando disponible en el interprete de comandos. En este ejemplo, ejecutamos el comando ping. Cuando hayamos terminado, simplemente tecleamos "exit" para salir del contenedor. El contenedor continúa funcionando después de salir de la sesión interactiva.

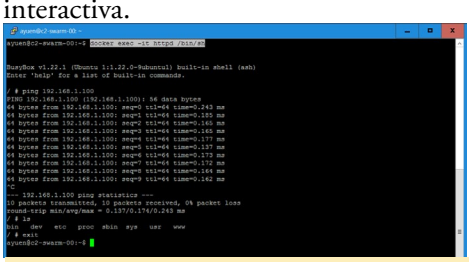


Figura 8 - Docker Exec

Podríamos haber iniciado el contenedor en modo interactivo cuando lo iniciamos. Por ejemplo, en lugar de utilizar el siguiente comando:

```
$ docker run -d -p 80:80 --name
httpd \
  mrdreambot/arm64-busybox-httpd
```

Podríamos haber usado este:

```
$ docker run -it -rm \
  mrdreambot/arm64-busybox-httpd
bin/sh
```

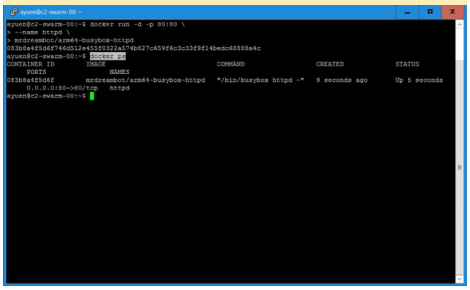
Las opciones utilizadas son:

-it significa modo tty interactivo

-rm significa eliminar el contenedor cuando salimos del intérprete.

/bin/sh significa reemplazar el punto de entrada por defecto (iniciando el servidor httpd) con el comando /bin/sh.

Figura 9 – Resultado del comando PS



Deteniendo y eliminando los contenedores activos

Para listar los contenedores activos, ejecuta el siguiente comando:

```
$ docker ps
```

Para detener y eliminar el contenedor httpd que iniciamos, utiliza los siguientes comandos:

```
$ docker stop httpd
$ docker rm httpd
```

A veces, cuando ejecutas un contenedor en una sesión de interacción y te sales, no verás el contenedor con el comando "docker ps". Tienes que usar el siguiente comando:

```
$ docker ps -a
```

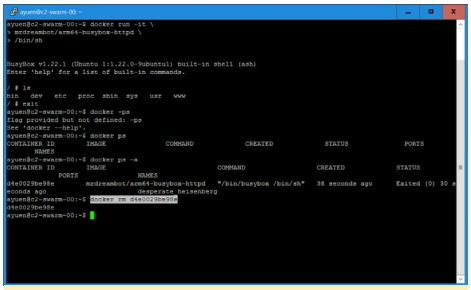
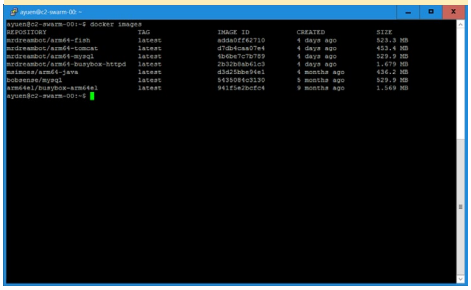


Figura 10 - Resultados de comando docker ps -a y docker rm

No proporcione ningún nombre a la se-sión interactiva. Tengo que identificar la sesión y usar el comando "docker rm" tal y como se muestra donde d4e-0029be98e es el ID del contenedor identificado en el comando "docker ps -a".

Figura 11 - Imágenes de Docker



Gestionando imágenes

Puede listar las imágenes disponibles en tu host Docker usando el comando "docker images" como se muestra en la Figura 11. Como ya he comentado, si no tienes la imagen en tu host Docker al iniciar un contenedor, Docker intentará descargarla desde Docker Hub. Sin embargo, puedes descargar la imagen de antemano con el comando "docker pull":

```
$ docker pull mrdreambot/arm64-
busybox-httpd
```

Puede eliminar una imagen de su host Docker utilizando el siguiente comando, donde adda0ff62710 es el ID de la imagen que aparece con el comando "docker images":

```
$ docker rmi adda0ff62710
```

En la Figura 11, adda0ff62710 hace referencia a la imagen mrdreambot/arm64-fish.

Buscando imágenes

La Figura 12 muestra dos ejemplos de búsqueda de imágenes arm64. El primer comando busca todas las imágenes con "arm64" en el nombre de la imagen y limita el resultado a un máximo de 10.

El segundo comando busca todas las imágenes con "arm64" en su nombre y que tengan una valoración de 5 estrellas o superior. Puede utilizar la opción -help para explorar las otras opciones que tiene el comando de búsqueda:

```
$ docker search --help
```

También puede hacer una búsqueda usando tu navegador y accediendo a http://hub.docker.com. Puedes obtener más información sobre cómo utilizar la imagen que te interesa en Docker Hub, tal y como se muestra en la Figura 13.

Figura 12 - Búsqueda de Docker

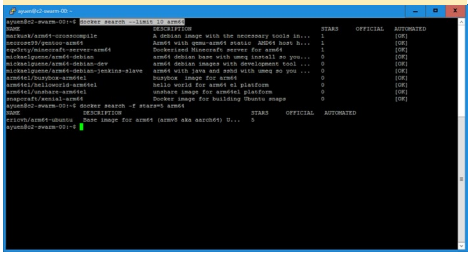
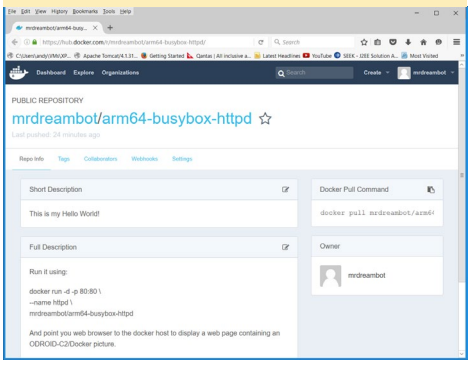


Figura 13 - Detalles de la imagen



Gestionando el almacenamiento persistente

Si guardas datos dentro de tu contenedor Docker, los datos desaparecerán una vez que el contenedor se elimine utilizando el comando "docker rm". Hay diferentes maneras de hacer que tus datos sean permanentes, como son el uso de volúmenes de datos o contenedores de datos. Sin embargo, la manera más sencilla es usar la opción "-v" como se muestra a continuación:

```
$ docker run -d -p 3306:3306 \
 -- name mysql \
 -e MYSQL_USER=fishuser \
 -e MYSQL_PASSWORD=fish456 \
 -e MYSQL_DATABASE=fish \
 -v /media/sata/fish-mysql:/u01/
my3306/data \
 mrdreambot/arm64-mysql
```

-e MYSQL_USER=fishuser ajusta la variable de entorno para decirle al contenedor que configure el usuario de MySQL a fishuser

-e MYSQL_PASSWORD=fish456 ajusta la variable de entorno para decirle al contenedor que configure la contraseña de MySQL a fish456

-e MYSQL_DATABASE=fish ajusta la variable de entorno para decirle al contenedor que configure la base de datos para usar fish

-v /media/sata/fish-mysql:/u01/my3306/data vincula el volumen /media/sata/fish-mysql al directorio :/u01/my3306/data del contenedor que es el directorio de la base de datos MySQL. Esto significa que MySQL guardará todos los datos en el directorio de tu host /media/sata/fish-mysql.

Aunque hemos facilitado el almacenamiento a la base de datos, todavía no hemos iniciado el contenido de la base de datos. En la siguiente sección detallaremos como hacerlo.

Usando múltiples contenedores Docker

En esta sección, voy a poner en práctica un escenario muy común en el que una aplicación tiene un front-end web y un back-end con una base de datos MySQL. Esto significa que el front-end web y la base de datos se ejecutan en tu contenedor. En este tutorial usaremos una aplicación de ejemplo WEB4J llamada "Fish and Chips Club". A partir de ahora, voy a referirme a esta aplicación como "Fish". Esta aplicación permite:

- editar miembros del club
- editar restaurantes locales
- editar las valoraciones de cada restaurante
- añadir nuevos almuerzos (un restaurante especifico en un día concreto)
- RSVP para el próximo almuerzo
- interactuar utilizando un sencillo foro de discusión
- elaborar informes sencillos
- realizar búsquedas simples

Fish utiliza 3 bases de datos que se ejecutan en MySQL. Puede obtener más información sobre cómo configurar esta aplicación en http://bit.ly/2eHmxOW.

Ejecutando la aplicación Fish

Para iniciar el contenedor Fish, usa el siguiente comando:

```
$ docker run -d \
 -p 8080:8080 \
 --name fish \
 -e MYSQL_SERVER=192.168.1.100 \
 -e MYSQL_PORT=3306 \
 mrdreambot/arm64-fish
```

La opción -e MYSQL_SERVER=192.168.1.100 ajusta la variable de entorno para indicar al contenedor el nombre/dirección IP del servidor MySQL

-e MYSQL_PORT=3306 ajusta la variable de entorno para indicar al contenedor el puerto que se utilizará para acceder al servidor MySQL

Ten en cuenta que aún no hemos creado las bases de datos de fish con contenido. Lo vamos a hacer en la siguiente sección.

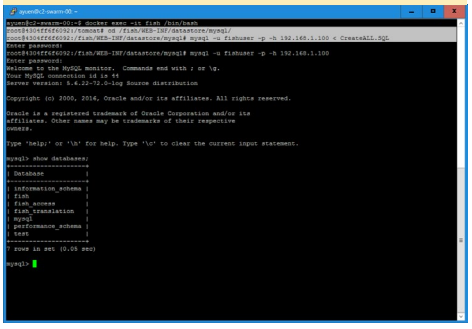
Configurando la base de datos

Para configurar las bases de datos de Fish, utiliza los siguientes comandos:

```
$ docker exec -it fish /bin/bash
$ cd /fish/WEB-INF/datastore/
mysql/
$ mysql -u fishuser -p -h
192.168.1.100 < CreateALL.SQL
$ mysql -u fishuser -p -h
192.168.1.100
$ show databases;
$ exit
$ exit
```

El script SQL para configurar las bases de datos Fish está en el directorio /fish/WEB-INF/datastore/mysql/ del contenedor Fish. Vamos a acceder al contenedor fish activo usando "docker exec" y procederemos a ejecutar el cliente MySQL para iniciar la base de datos MySQL que se ejecuta en un contenedor distinto. Después de eso, usamos el cliente MySQL nuevamente para verificar que las bases de datos han sido creadas usando el comando "show databases;", tal y como se muestra en la Figura 14.

Figura I4 - Crear la base de datos



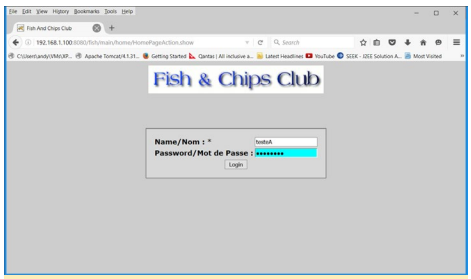


Figura I5 – Accediendo a Fish and Chips

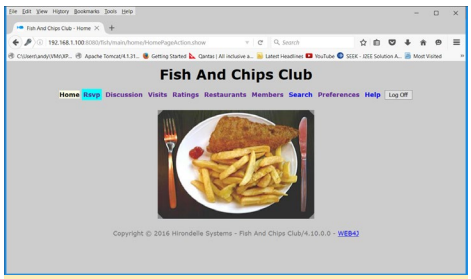


Figura I6 - Fish and Chips club

Probrando "Fish"

Ahora que hemos creado las bases de datos necesarias para Fish con el contenido adecuado, podemos a través de un navegador web acceder a la aplicación Fish introduciendo http://192.168.1.100:8080/fish. Reemplaza 192.168.1.100 por la dirección IP de tu ODROID-C2. Se te pedirá que inicies sesión. Utiliza "testeA" y "testtest" como nombre de usuario y contraseña respectivamente. Tras iniciar sesión, verás la página principal de Fish.

Creando una imagen Docker

Hasta ahora, hemos utilizado imágenes creadas por mí. ¿Cómo creamos una imagen desde el principio? Aquí tienes una guía rápida de cómo hacerlo. En primer lugar, clona mi mrdreambot/arm64-busybox-httpd de Github en http://bit.ly/2eWnLdb. Aquí está el contenido del Dockerfile que le dice a Docker cómo se van a crear las imágenes:

```
FROM arm64el/busybox-arm64el
MAINTAINER MrDreamBot
COPY www /www
ENTRYPOINT ["bin/busybox"]
CMD ["httpd, "-f", "-p", "80",
"-h", "/www"]
```

FROM - especifica que estas imágenes están basadas en las imágenes arm64el/busybox-arm64el

MAINTAINER - especifica el autor del Dockerfile

COPY - copia el directorio www del directorio actual al directorio /www de la imagen

ENTRYPOINT - configura el comando y el argumento por defecto para iniciar el contenedor

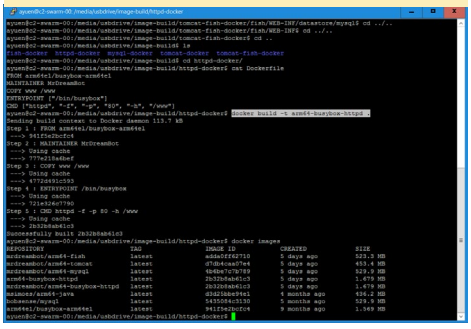
CMD - ajusta los valores adicionales por defecto que es probable que hayan cambiado.

Si mrdreambot/arm64-busyboxhttpd sigue apareciendo cuando utilizas el comando "docker images". Utiliza el comando "docker rmi" para retirar las imágenes antes de realizar el siguiente paso. A continuación, cambia al directorio donde clonaste mi proyecto y usa el siguiente comando, tal y como se muestra en la Figura 17:

```
$ docker build -t arm64-busybox-
httpd .
$ docker images
```

Ahora has creado tu primera imagen de Docker llamada arm64-busyboxhttpd, que se puede explotar del

Figura I7 – Crear la imagen



mismo modo que el que se muestra en la sección "Ejecutando tu primer contenedor". Simplemente reemplaza "mrdreambot/arm64-busybox-httpd" por "arm64-busybox-httpd" en el comando "docker run" y dirige tu navegador hacia ella para ver la misma imagen en ODROID-C2.

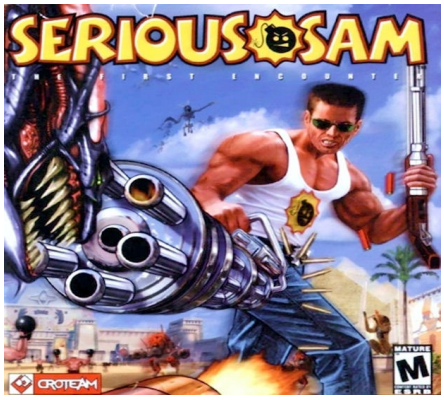
¿Qué es lo próximo?

A lo largo de este tutorial, he descrito en líneas generales todos los comandos clásicos de Docker que necesitas conocer para ejecutar y administrar aplicaciones que se ejecuten en contenedores. Los comandos de Docker que te he mostrado sólo funcionarán en tu host Docker local, que en este caso es tu ODROID-C2. Te darás cuenta muy pronto que hay un límite en el número de contenedores que puedes ejecutar en una única máquina. ¿Qué pasa con las empresas que utilizan Docker en un entorno de producción? Seguramente una sola máquina no será capaz de ejecutar toda su carga de trabajo. Aquí es donde entra en acción el modo Swarm de Docker. El modo Swarm es nuevo en Docker 1.12. Cuenta con un motor de organización y planificación integrado, lo que significa que, en este contexto, gestiona el algoritmo de agrupamiento, la programación de la carga de trabajo y la administración del estado. El algoritmo de agrupamiento hace referencia a un conjunto de máquinas que trabajan al mismo tiempo actuando como una sola máquina. La programación y la administración del estado permiten decidir dónde ejecutar los contenedores entre las máquinas que integran un clúster y cuántas réplicas de contenedores deberían ejecutarse. Con anterioridad a Docker 1.12, tenías que llevar a cabo un extenso trabajo de configuración antes de poder conseguir organizar Docker. En la Parte 2, te mostraré cómo usar los comandos del modo swarm para poner en práctica la planificación de Docker.

JUEGOS LINUX

LOGRAR EJECUTAR SERIOUS CON EL SERIOUS-ENGINE

por Tobias Schaaf



En este artículo, quisiero hablar de un tipo que es casi tan renegado como el propio Duke Nukem Su nombre es Sam, ¡y es muy serio! Hace un tiempo, observé que el usuario @ptit-Seb de los foros OpenPandora estaba trabajando en uno de sus muchos e impresionantes trabajos de migración de juegos. Esta vez, se trataba de "Serious-Engine", que es un motor de código abierto para el juego Serious Sam - The First Encounter y Serious Sam: The Second Encounter. Recuerdo muy bien este juego, pasé muchas horas con un amigo luchando contra oleadas y oleadas de monstruos. Inmediatamente empecé a compilar los juegos para la plataforma ODROID con un éxito moderado. Conseguí que funcionaran, pero como no había instalador y la estructura de los archivos del juego y las librerías era algo extrañas, abandoné temporalmente el proyecto. @ptitSeb siguió mejorando su versión, así como GLshim que él utilizó para ejecutar el motor, y ahora el juego se ejecuta muy bien. Dedique nuevamente un tiempo a compilar y probar el juego, y finalmente fue capaz de crear un instalador con el que tener en cuenta los diferentes requisitos del juego, así como facilitar a los usuarios el poder añadir los datos del juego que faltasen.

Instalación

Como de costumbre, puede instalar este juego desde mi repositorio, los pasos de instalación están detallados en http://bit.ly/2eOG92v. Serious Sam está en mi lista de paquetes principal/jessie. Ambos juegos: Serious Sam - The First Encounter (TFE) y Serious Sam – The Second Encounter (TSE) se puede instalar por separado dependiendo del que prefieras:

```
$ apt-get install ssam-tfe-odroid
$ apt-get install ssam-tse-odroid
```

Serious-Engine requiere OpenGL y como ODROIDs no tienen OpenGL pero si OpenGL ES, tenemos que usar GLshim para ejecutar el juego, que lo facilita también @ptitseb. Además necesitarás los datos del juego original, específicamente las carpetas "Data", "Levels" y "Demo", así como todos los archivos ".gro". Estos archivos deben colocarse en la carpeta del juego, que se encuentra en la carpeta de inicio del usuario actual y que está nombrada con ".tfe" o ".tse". Copia tus archivos del juego en esta carpeta y estarás listo para jugar.

Problemas conocidos

Aunque el juego funciona bastante bien, me encontré con algunos problemas que espero que se puedan solucionar en algún momento en el futuro. Por ejemplo, parece que hay algunos errores gráficos con las placas Exynos (ODROID-X, X2, U2, U3, XU3 y XU4) que muestran colores extraños en diferentes lugares. Estos problemas no aparecen en todas partes y posiblemente se puedan ignorar, pero son un poco

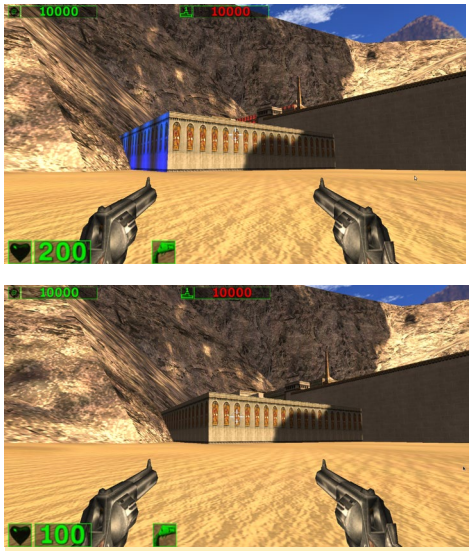


Figura 1 - ODROID XU3 / XU4 (arriba) vs ODROID C2 (abajo), podemos observar algunos de los fallos en el color que se pueden apreciar en los dispositivos Exynos

molestos. No se aprecian los mismos fallos en el C2, por lo que aparentemente el tema puede estar relacionado con los drives Exynos. Aunque no son bonitos, estos fallos no te suponen un obstáculo para poder jugar al juego. No aparecen con demasiada frecuencia y en niveles posteriores desaparecen por completo, de modo que estos errores se pueden pasar por alto con total seguridad.

También descubrí que la música no funciona por ahora, pero creo que es un problema que se puede solucionar, y que podría estar resuelto para cuando se publique este artículo. Además, parece

Figuras 2 y 3 - Serious Sam ofrece gran cantidad de armas y monstruos

Figura 4 y 5 - Me encanta los excelentes efectos de disparo en Serious Sam

que los modos multijugador todavía no funcionan. No puedes unirte a una partida por Internet, a pesar de que hay un montón de servidores disponibles. Al parecer, incluso las partidas en LAN no funcionan correctamente, así que espero que esto también se pueda solucionar pronto.

También intenté dividir la pantalla, parece que funciona. Es posible jugar con hasta cuatro personas con diferentes mandos, o con un único jugador usando el ratón y el teclado. Sin embargo, el C2 no es lo suficientemente potente como para manejar dos jugadores simultáneamente y en el XU4, observé algunos fallos gráficos que hacen que la imagen empiece a parpadear, parece ser un problema de sincronización vertical. Espero que en las partidas LAN esto se pueda solucionar, ya que este juego es impresionante en modo multijugador.

Parece que también hay otro problema en Serious Sam - The Second Encounter, donde te deslizas por debajo de la superficie y te encuentras atrapado entre el nivel y el suelo. Ocurre al azar y sólo en algunos lugares. Si te sucede, no puedes hacer nada excepto recargar el juego y reiniciarlo desde el último punto guardado. @ptitSeb está trabajando duro para solucionar estos problemas, actualizaré Serious Sam y GLshim cuando haya correcciones disponibles.

Jugabilidad

Con todos estos problemas te estarás preguntando, ¿Realimente funciona? Bueno, parece que casi todo lo demás si. Puedes jugar a Serious Sam – The First Encounter y Serious Sam – The Second Encounter en la modalidad de jugador único con todos sus cosas buenas y malas. Realmente es un shooter único que desafía todos los tipos de realismo. Diriges un ejército de un solo hombre, matando a cientos y cientos de monstruos. El juego se ve genial y se ejecuta muy bien en los ODROIDs.

Como puedes ver en las capturas de pantalla, todavía puedes jugar y disfrutar del juego. Diviértete jugando y vigila los foros para actualizaciones y correcciones de errores.

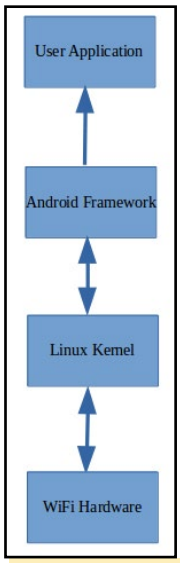
DESARROLLO ANDROID

PILA WIFI DE ANDROID

por Nanik Tolaram

No sólo son dispositivos Android realmente potente que vienen repletos de características, sino que también pueden ser portátiles y fáciles de llevar. Pero ¿De qué sirve un dispositivo sin conexión a Internet? Por supuesto, la portabilidad significa no depender de los cables, lo que nos lleva a dos formas principales de conseguir que nuestro dispositivo esté siempre conectado: a través de Wi-Fi o mediante una conexión por móvil. Todos los dispositivos Android tienen la funció de Wi-Fi integrada, y en este artículo vamos a echar un vistazo a cómo funciona internamente la conectividad Wi-Fi dentro del sistema operativo. El código fuente que he utilizado en este artículo está basado en la versión de desarrollo android-5.1.1_r38 del Android Open Source Project (AOSP).

Esquema de alto nivel



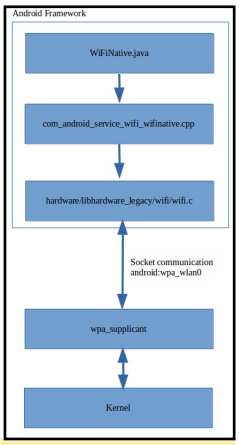
Esquema general de la pila de alto nivel de Android

Comencemos por echar un vistazo a esta Figura. Muestra una interacción de alto nivel entre las diferentes pilas del sistema dentro de Android. En la capa superior es donde se ejecutan nuestras aplicaciones como YouTube y Twitter. Para facilitar el desarrollo de aplicaciones, Android utiliza un entorno de trabajo que ayuda a que estas aplicaciones se comuniquen con la tecnología a nivel de hardware y el kernel que hace que funcione nuestra conexión a Internet. En esta segunda capa es donde vamos a analizar la pila Wi-Fi para Android y veremos cómo ésta funciona entre el kernel Linux y nuestras aplicaciones.

Entender la wpa_supplicant

En nuestro anterior artículo, analizamos el HAL (Hardware Abstraction Layer) dentro de Android para entender cómo las aplicaciones utilizan este entorno de trabajo para comunicarse con el hardware dentro de nuestros dispositivos. Normalmente la mayoría del software aprovecha esta capa, pero el Wi-Fi no lo hace. Esto es porque utiliza una pila de bajo nivel de código abierto para soportar la comunicación software-hardware llamada wpa_supplicant.

El entorno de trabajo de Android utiliza wpa_supplicant como una forma para comunicarse con el kernel de Linux, al igual que ocurre con muchos otros sistemas operativos basados en Linux y Unix. Esto se logra utilizando la librería cliente wpa_supplicant para comunicarse a través de una conexión socket con el demonio wpa_supplicant ejecutado en el dispositivo. Tal y como muestra esta Figura, los comandos se inician desde una aplicación en el entorno de trabajo de Android que utiliza la conexión socket hacía el demonio para transmitir comandos WiFi al propio hardware. El demonio ayuda a garantizar que estos comandos, como la activación y desactivación de la señal Wi-Fi, sean entendidos por el kernel de Linux.



Un análisis detallado de la pila Wi-Fi

El wpa_supplicant se inicia durante el proceso de arranque de Android, tal y como se observa en la imagen de la siguiente página. Este fragmento fue cogido de http://bit.ly/2eNRphu.

```
service wpa_supplicant /system/bin/wpa_supplicant \
    -iwlan0 -Dnl80211,wext -c/data/misc/wifi/wpa_supplicant.conf \
    -e/data/misc/wifi/entropy.bin \
    -O/data/misc/wifi/sockets \
    -g@android:wpa_wlan0
    #   we will start as root and wpa_supplicant will switch to user wifi
    #   after setting up the capabilities required for WEXT
    #   user wifi
    #   group wifi inet keystore
    class main
    socket wpa_wlan0 dgram 660 wifi wifi
    disabled
    oneshot
```

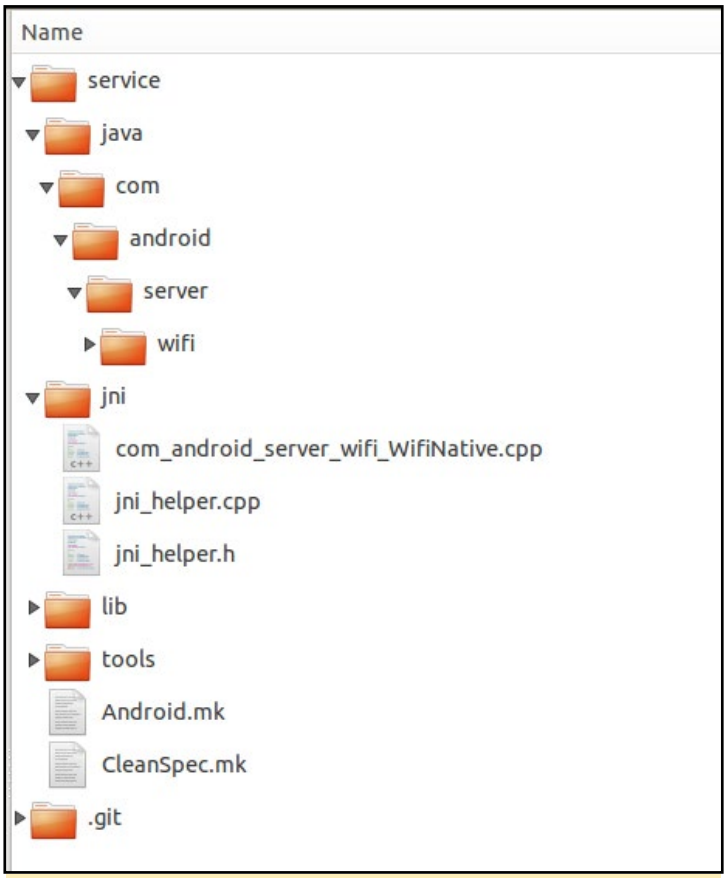
El proceso de inicio del demonio wpa_supplicant

Entorno de trabajo interno

Las aplicaciones de usuario Android tienen acceso al hardware Wi-Fi bajo la capa del kernel Linux a través del entorno de trabajo mediante el acceso al servicio Context.WIFI_SERVICE utilizando el objeto Context:

```
$ Context.getSystemService(Context.WIFI_SERVICE)
```

El usuario lanzará una instancia de WiFiManager que permitirá a los usuarios acceder a los servicios Wi-Fi, como su conexión activa actual, poder volver a asociarse con la conexión y muchas otras funciones. Puedes ver la implementación del gestor "real" del Wi-Fi en la siguiente figura dentro del directorio works/opt/net/wifi directory.



El directorio frameworks/opt/net/wifi

Si está interesado en conocer el código el entorno de trabajo que reside en cada una de estas carpetas, puede revisar la información que aparece en la Tabla 1.

frameworks/opt/net/wifi/service/java/com/android/server	Contains Java code that is responsible for exposing the API (WifiManager) for user application to access wifi specific functionality, Storing wifi state management, wifi security, etc
frameworks/opt/net/wifi/service/jni	Folder containing Android source code interfacing with the wpa_supplicant library

Tabla 1 – El código del directorio frameworks/opt/net/wifi

El código de bajo nivel que se comunica con el wpa_supplicant se empaqueta dentro del archivo binario libwifi-service.so, que se puede ver en el Makefile de Android.mk tal y como se muestra en la figura de la izquierda.

```
include $(CLEAR_VARS)

LOCAL_REQUIRED_MODULES := libandroid_runtime libhardware_legacy

LOCAL_CFLAGS += -Wno-unused-parameter -Wno-int-to-pointer-cast
LOCAL_CFLAGS += -Wno-maybe-uninitialized -Wno-parentheses
LOCAL_CPPFLAGS += -Wno-conversion-null

LOCAL_C_INCLUDES += \
        $(JNI_H_INCLUDE) \
        $(call include-path-for, libhardware)/hardware \
        $(call include-path-for, libhardware_legacy)/hardware_legacy \
        libcore/include

LOCAL_SHARED_LIBRARIES += \
        libnativehelper \
        libcutils \
        libutils \
        libhardware \
        libhardware_legacy \
        libandroid_runtime \
    libnl

LOCAL_STATIC_LIBRARIES += $(LIB_WIFI_HAL)

LOCAL_SRC_FILES := \
        jni/com_android_server_wifi_WifiNative.cpp \
        jni/jni_helper.cpp

LOCAL_MODULE := libwifi-service

include $(BUILD_SHARED_LIBRARY)
```

Un vistazo a Android.mk

Como puede ver, el archivo libwifi-service.so utiliza com_android_server_wifi_WifiNative.cpp, el cual está cargado por el entorno de trabajo tal y como muestra la Figura de la izquierda, durante el tiempo que se está configurando la clase.

```
public class WifiNative {

    private static boolean DBG = false;
    private final String mTAG;
    private static final int DEFAULT_GROUP_OWNER_INTENT     = 6;

    static final int BLUETOOTH_COEXISTENCE_MODE_ENABLED     = 0;
    static final int BLUETOOTH_COEXISTENCE_MODE_DISABLED    = 1;
    static final int BLUETOOTH_COEXISTENCE_MODE_SENSE       = 2;

    static final int SCAN_WITHOUT_CONNECTION_SETUP          = 1;
    static final int SCAN_WITH_CONNECTION_SETUP             = 2;

    // Hold this lock before calling supplicant - it is required to
    // mutually exclude access from Wifi and P2p state machines
    static final Object mLock = new Object();

    public final String mInterfaceName;
    public final String mInterfacePrefix;

    private boolean mSuspendOptEnabled = false;

    /* Register native functions */

    static {
        /* Native functions are defined in libwifi-service.so */
        System.loadLibrary("wifi-service");
        registerNatives();
    }
```

Un vistazo a WifiNative.java

Gestión del estado

La pila Wi-Fi rastrea cualquier cambio en la conexión Wi-Fi y esto lo hace internamente usando la gestión del estado. Manteniendo pestañas con diferentes estados del Wi-Fi nos permite que el entorno de trabajo reaccione en consecuencia y ofrezca la posibilidad de informar a la aplicación del usuario a través del objetivo de transmisión de datos. A continuación detallamos las diferentes clases que usamos internamente para entender la gestión del estado del Wi-Fi.

Clases

DefaultState
InitialState
DriverStoppingState
DriverStartingState
DriverStartedState
DriverStoppedState
ScanmodeState
SupplicantStartingState

SupplicantStartedState
SupplicantStoppingState
VerifyingLinkState
RoamingState
L2ConnectedState
WaitForP2PDisableState
SoftAPRunningState
WPSRunningState
SoftAPStartedState
ConnectedState
ObtainingIPState
DisconnectingState
DisconnectedState
ConnectModeState
TetheringState
TetheredState
TetheringState
UntheteringState

Lista de objetivos de la gestión del estado del Wifi

Los objetivos son el elemento vital de las aplicaciones de Android, internamente el entorno de trabajo usa cada objetivo para comunicar estados wifi a las aplicaciones del usuario. Echemos un vistazo a un ejemplo de cómo se utiliza el objetivo internamente para informar a las aplicaciones de usuario sobre algún estado. El ejemplo más fácil es cuando las aplicaciones quieren ser informadas sobre el estado de Wi-Fi (activado/desactivado) para que puedan tomar alguna acción en consecuencia. Esto se hace utilizando la siguiente declaración <intent> en AndroidManifest.xml.

```
<receiver android:name=".WifiReceiver">
   <intent-filter>
     <action android:name="android.net.wifi.WIFI_
STATE_CHANGED" />
   </intent-filter>
</receiver>
```

Dentro de la aplicación habrá una clase que amplíe BroadcastReceiver así:

```
public class WifiReceiver extends BroadcastReceiver {
    …
    @Override
    public void onReceive(final Context context, final
Intent intent) {
        int wifiState = intent.getIntExtra(WifiManager.
EXTRA_WIFI_STATE, -1);
        if (WifiManager.WIFI_STATE_CHANGED_ACTION.
equals(intent.getAction())
                    && WifiManager.WIFI_STATE_ENABLED ==
wifiState) {
                …
            }
    }
```

La aplicación ya está lista para recibir el objetivo del entorno de trabajo cuando el wifi cambie de estado. Dentro del entorno de trabajo este objetivo se transmite en la clase WifiStateMachine.java.

```
private void setWifiState(int wifiState) {
    final int previousWifiState = mWifiState.get();

    try {
        if (wifiState == WIFI_STATE_ENABLED) {
            mBatteryStats.noteWifiOn();
        } else if (wifiState == WIFI_STATE_DISABLED) {
            mBatteryStats.noteWifiOff();
        }
    } catch (RemoteException e) {
        loge("Failed to note battery stats in wifi");
    }

    mWifiState.set(wifiState);

    if (DBG) log("setWifiState: " + syncGetWifiStateByName());

    final Intent intent = new Intent(WifiManager.WIFI_STATE_CHANGED_ACTION);
    intent.addFlags(Intent.FLAG_RECEIVER_REGISTERED_ONLY_BEFORE_BOOT);
    intent.putExtra(WifiManager.EXTRA_WIFI_STATE, wifiState);
    intent.putExtra(WifiManager.EXTRA_PREVIOUS_WIFI_STATE, previousWifiState);
    mContext.sendStickyBroadcastAsUser(intent, UserHandle.ALL);
}
```

Enviando WIFI_STATE_CHANGED_ACTION

Internamente, existe más código antes de llamar al setWifiState (..) pero este es el punto donde el usuario obtendrá información sobre el estado actual de Wi-Fi. Espero que esto te haya dado una idea de cómo funciona la pila Wi-Fi y cómo interactúan tus aplicaciones con ella.

MYTHTV

EJECUTAR UNA APLICACION DE ENTRETENIMIENTO EN CASA DE CODIGO ABIERTO EN TU ODROID-C2

por @WebMaka

Utilizo varios ODROID-C2 en mis televisores como front-ends para ver secuencias de video en vivo, usando MythTV como software principal. El back-end es un ordenador estándar que ejecuta Ubuntu 16.04.1 LTS y usando un HDHomeRun PRIME con una CableCard y un sintonizador como medio para convertir mi servicio de cable digital en una señal por streaming me permite ver videos de forma remota. Puesto que configurar todo esto la primera vez puede llevar mucho tiempo, aquí tienes una serie de consejos prácticos que te ayudarán con tu propia instalación. Esta guía está escrita para la última versión LTS de Ubuntu, la 16.04.1 LTS, aunque funciona también con la 14.04 LTS y debería funcionar igualmente en cualquier distribución derivada de Debian.

Acerca de DRM

La DRM es una pesadilla en todas partes para los usuarios de TV en vivo. Espero que este sea tu mayor problema y si estás intentando apartar de la televisión a tu esposo/a o tus padres, espero que la disputa simplemente sea el no poder ver los canales o el contenido DRMed.

Tenga en cuenta que esto significa que MythTV no puede y no funcionará con ninguna programación de TV en vivo marcada como "protected/copy-once" o "protected/no-copy. Intencionadamente, no hay forma de evitar esto con algún tipo de software de código abierto. Este planteamiento está basado en la obligación de usar un hardware y software "oficial", del que dispones para gastar dinero mes tras mes si quieres utilizarlo, bajo la premisa de que esto protege el contenido de la redistribución.

Algunas compañías por cable marcan casi toda su oferta como "copy-once", pero otras sólo ofrecen canales premium de pago como HBO o Cinemax. Algunos canales que, en mi opinión, no deberían estar marcados como "copy-once", a menudo están así debido a la propia demanda del canal y no por la propia compañía por cable. Si tu compañía por cable ofrece todo o casi todo de forma muy restringida, no la tomes con HDHomeRuns o con cualquier otro equipo propietario de consumo. Tendrás que continuar usando el sistema por cable o simplemente darte de baja del servicio de televisión por cable.

El HDHomeRun Prime puede ver contenido protegido con el software apropiado, pero si deseas ver y transmitir contenido protegido por DRM, la única forma segura de hacerlo es utilizando el Windows Media Center en Windows. Ten en cuenta que sólo Windows 7 incluye Windows Media Center de forma gratuita. SiliconDust supuestamente tenía una aplicación para Android que permitía ver el contenido "copy-once", pero la aplicación actual no lo hace de forma coherente. He oído que es posible ver el contenido DRMed desde un HDHR Prime con un Nvidia Shield, pero no lo he intentado.

Configuración

Estos son los pasos necesarios para usar MythTV con tu ODROID:

1. Instalar y actualizar Ubuntu 16.04 LTS
2. Instalar el software driver HDHomeRun en tu sistema Linux
3. Instalar y configurar el backend de Myth-TV en tu sistema Linux
4. Configurar los clientes/frontends en tu ODROID
5. Visualizar la TV en vivo junto con todo lo que pueda descargar/retransmitir por Internet

Instalalar y actualizar Ubuntu

Si estás utilizando la última versión 16.04 LTS de Ubuntu, empieza usando los siguientes comandos para que todo esté actualizado:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Si está usando una versión anterior, ten en cuenta que la actualización del sistema operativo requerirá más tiempo y es más arriesgado que simplemente reinstalar a partir de la última imagen. Estas instrucciones deben funcionar también con otras distribuciones basadas en Debian, aunque tu experiencia podría ser algo diferente.

Instalar el driver HDHomeRun

Para instalar la herramienta de configuración y las librerías de HDHomeRun, utiliza el siguiente comando:

```
$ sudo apt-get install libhdhome-
run hdhomerun-config
```

A continuación, instala el driver adecuado. Lo más fácil es coger el driver DVB con el paquete Debian que fue publicado en GitHub por @h0tw1r3:

```
$ git clone https://github.com/
h0tw1r3/dvbhdhomerun
$ cd dvbhdhomerun
$ dpkg-buildpackage -b
```

Si esto falla con un error de dependencia, necesitarás ejecutar "sudo apt-get install" para cualquier paquete que falte, como dkms. Suponiendo que el comando buildpackage se haya completado correctamente, instala los paquetes de desarrollo:

```
$ cd ..
$ dpkg -i *hdhomerun*.deb
```

Una vez hecho esto, deberías poder lanzar la GUI de configuración de HDHomeRun y tener una lista con los tres sintonizadores de Prime. Si no detecta el HDHR Prime, corrige esto antes de continuar, ya que puede que te hayas saltado un paso o que hayas tenido un fallo que has pasado por alto.

Instalar y configurar el backend MythTV

Escribe el siguiente comando para instalar el paquete MythTV:

```
$ sudo apt-get install mythtv
```

Como es de esperar, en una nueva instalación de Ubuntu serán instaladas bastantes dependencias, así que dejar que el sistema haga lo que tiene que hacer. A continuación, haz clic en el botón Ubuntu en la parte superior de la barra y luego, haga clic en MythTV Backend Setup. Puede que necesites buscarlo en la primera fila de iconos, si no lo ves.

Si te solicita información para una conexión de base de datos, anota la configuración para el nombre de la base de datos, el nombre de usuario y la contraseña, y a continuación cierre la aplicación de configuración del back-end. Tendrás que añadir la base de datos a MySQL, crear la cuenta de usuario que MythTV utilizará y darle permisos completos en la base de datos que utilizará MythTV. No te limites a facilitar el nombre de usuario y contraseña "root" de MySQL, ya que es muy poco seguro. En su lugar, utiliza una herramienta de gestión MySQL para crear un nuevo usuario para MythTV. A continuación, reinicia el programa de instalación de back-end. Si todo ha ido bien, no te debería solicitar datos de conexión a la base de datos. Si no, corrige esto antes de continuar.

Es importante proporcionar una dirección IP estática a la máquina que será el backend primario de MythTV. Necesitarás tener acceso a él con una IP fija, DHCP probablemente cambiará la dirección IP siempre que el reinicies el router. Consulte la documentación de su router para saber cómo asignar IPs estáticas. Haz esto antes de continuar si la máquina tiene actualmente una dirección IP dinámica/asignada por DHCP.

Configurar la dirección IP

Una vez que la herramienta de configuración del backend de MythTV esté satisfecha con la configuración de MySQL y pueda conectarse a la base de datos, deberías ver una serie de opciones con "General" como primera opción. En "Host Address Backend Setup" es donde empiezas. Modifica "IPv4 Address" por la dicción IP estática. No dejes 127.0.0.1, o sólo aceptará conexiones de clientes desde localhost, ignorando el resto de clientes de la red.

Deja "IPv6 Address" en blanco a menos que esté utilizando IPv6. Si no lo haces, MythTV podría vincularse únicamente con una dirección IPv6 ignorando las conexiones IPv4. Se supone que no, pero a mí me ocurrió y hay gente que también le ha sucedido. A continuación, cambia "Security Pin (required)" a 0000 (para cualquier cliente), o un pin real si necesitas restringir el acceso o MythTV no aceptará las conexiones.

En la parte "Master Backend", modifica "IP address" para que coincida con la IP estática, o de nuevo sólo usará localhost para las conexiones desde los backends esclavos. Mantén presionando "next" hasta que regreses al menú principal. Para la configuración inicial, no necesitará cambiar nada más en las opciones generales.

Configurar las tarjetas de captura

Va a la sección "Capture Cards". Lo haremos tres veces, una por cada sintonizador en el HDHR Prime. Primero, haz clic en "New Capture Card", o usa las teclas de flecha para seleccionar y pulsa Enter. Para la primera casilla, "DVB Device", abre la lista y elije la opción HDHomeRun. Si no la tienes, te perdistes algo en el primer paso, así que retrocede en consecuencia. Si HDHomeRun no puede "ver" tu HDHR Prime, Myth-TV tampoco. En "Available Devices", abre la lista y selecciona el primer sintonizador. Será un código ID para el HDHR seguido de "-0", por ejemplo, 1234ABCD-0. "Tuner" debe ser "0" por razones obvias. Haz clic en "Recorder Options".

Aumenta "Signal timeout (ms)" a "10000" (lectura: añade cero al final) y aumenta "Tuning timeout (ms)" a "30000" (ver anterior). Esto reducirá los tiempos de espera de los clientes lentos o durante sobrecargas en redes no gigabit concurridas. Reduce "Max recordings" a "1", ya que el HDHR Prime solo puede soportar una acción a la vez con cualquier sintonizador.

Haz clic en "Next" y "Finish" hasta que veas la lista de tarjetas de captura, y repite la configuración para los otros dos sintonizadores en el HDHR Prime. Después, pulsa la tecla Escape para volver al menú principal.

Configuración de las fuentes de vídeo

En el menú principal, dirigete a "Video Sources". Aquí es donde configurás tu lista de canales y guía. Asigna a la guía un nombre en "Video Source name".

Si tiene una suscripción activa a SchedulesDirect.org con datos del listado/guía, cambia "Listings Grabber" a la opción SchedulesDirect y rellena los espacios en blanco con el nombre de usuario y la contraseña. A continuación, haz clic en "Retrieve Lineups" para recuperar lo que tienes seleccionado en tu cuenta SchedulesDirect.org. Selecciona la entrada apropiada para "Data Direct lineup". Myth-TV obtendrá los datos del listado/guía. Sólo asegúrate de dejar "Perform EIT scan" sin marcar, ya que los datos de EIT sobrescribirán todos los datos de la guía SD.org y tus guías serán mucho menos amigables

Si no tiene una suscripción activa en SchedulesDirect.org, considera la posibilidad de gastar 25$ al año por una, ya que sus guías son muy útiles. Por lo menos, consigue una prueba de 7 días para ver si sus datos satisfacen tus necesidades. De lo contrario, si te encuentras en una zona para la cual no tienen listados, deja "Listores grabber" en " Transmitted guide only (EIT)" para intentar bajar la guía de la compañía por cable o cambiala a "xmltv Selections" y echa un vistazo a las opciones por si quiere usar otra configuración.

De nuevo haz clic en "Next" y "Finish" hasta que regrese a la lista de fuentes. Repite los pasos si deseas configurar más de una y presione Escape para volver al menú principal cuando finalices.

Conexiones de entrada

Dirígete a la sección "Input Connections", donde le indicas a MythTV que use los datos del programa, la frecuencia y el canal de la fuente para indicar a los sintonizadores qué sintonicen. También haremos esta parte tres veces, una vez por cada sintonizador, pero con un par de cambios sutiles pero importantes. Haz clic en el primer sintonizador HDHR de la lista o utiliza las teclas de flechas y la tecla Intro. He dejado "Display name" en blanco pero se puede cambiar por cualquier nombre siempre que sea único.

En "Video Source", selecciona el dispositivo que había configurado anteriormente, como tu cuenta SchedulesDirect.org. Con "Use quick tuning" no percibí ninguna diferencia en mi HDHR Prime, así que lo dejé en "Never".

Haz clic en el botón "Fetch channels from listings source" y presiona la tecla de flecha hacia abajo dos veces. Tardará unos segundos en buscar los datos del canal/frecuencia/programa, y sabrás que ha terminado cuando el control seleccionado salte de repente al botón "Next". Cuando lo haga, haga clic en "Next.

Este es un paso muy importante: "Schedule Order" y "Live TV Order" serán diferentes para cada sintonizador. El sintonizador 0 será "3" para "Schedule Order" y "1" para "Live TV Order", el sintonizador 1 será "2" para ambos, y el sintonizados 2 será "1" para "Schedule Order" y "3" para "Live TV order" 3 y 1, 2 y 2 y 1 y 3. Si no lo haces de esta forma, tendrás clientes sintonizado fuera de rango y probablemente el HDHR no tendrá acceso a todos sintonizadores porque MythTV pensará que está al final de la sintonización antes de que realmente pase por los tres. Además, si no les proporcionas una secuencia inversa, MythTV entregará el sintonizador 2 al primer cliente y el HDHR Prime le dirá a MythTV que está fuera de los sintonizadores disponibles. Me llevó un montón de tiempo descubrir esto, ya que no se mencionaba en ninguna parte, excepto en un único post del foro con el que topé por casualidad.

Dejé la configuración de grupo de entrada con los valores por defecto. Pulsa "Next" y "Finish" hasta que regreses a la lista de fuentes y repita los pasos si deseas configurar más de uno. Utiliza la tecla Escape para volver al menú principal cuando hayas terminado.

En el menú principal, vuelve a pulsar la tecla Escape. MythTV debería mostrar un cuadro de diálogo sobre cómo actualizar la guía si ha cambiado cualquier información de canal. Haz clic en OK o pulse Intro para descartar esto y cerrar el programa de configuración. Confirma que quieres ejecutar el backend de MythTV e introduce el nombre de usuario y la contraseña. A continuación, ejecuta "mythfilldatabase" para obtener la lista completa de canales y guía de programación. El backend aparece bastante rápido, lo cual puede probar navegando con un navegador a http://localhost:6544. La actualización de la guía tardará aproximadamente unos 20 minutos al ejecutarse la primera vez.

Configurar clientes

La mayoría de las personas que usan el backend de MythTV probablemente usaran Kodi para los clientes. Yo estoy ejecutando Kodi 16.1 (Jarvis) sobre una compilación personalizada de LibreELEC para el ODROID-C2 ya que el soporte de OpenELEC para el C2 no es tan sólido. Esto es muy probable que cambie cuando OpenELEC llegue a su primera versión completa 7.x, ya que el soporte para el C2 está en desarrollo.

Busca el plugin MyrTV PVR en Kodi navegando a System -> Settings -> Add-Ons -> Mis Add-Ons -> Addons PVR, una vez localizado selecciona "Config". Introduce la dirección IP de la máquina backend, activa el plugin seleccionando "Enable", luego reinicia Kodi. El plugin debería buscar la lista de canales/guía desde el backend y la sección "TV" debería aparecer en el menú principal.

Ver la TV en vivo

Desde tu dispositivo y entorno elegido, selecciona "TV", selecciona "Guide" y selecciona un canal/programa. Si todo funciona correctamente, deberías poder disfrutar de tu contenido multimedia en el ODROID-C2. Para comentarios, preguntas o sugerencias, visita el hilo original en http://bit.ly/2dvs3oQ.

ANDROID NOUGAT

IMPRESIONA A TUS AMIGOS CON LA ULTIMA VERSION DE ANDROID

por @voodik

Android Nougat, que fue lanzado oficialmente a principios de este año, introduce cambios muy importantes en el sistema operativo y su plataforma de desarrollo, incluyendo la posibilidad de mostrar múltiples aplicaciones en pantalla al mismo tiempo en una pantalla dividida, soporte para respuestas en línea a las notificaciones, así como un entorno Java basado en OpenJDK y soporte para la API de representación gráfica Vulkan, y actualizaciones "transparentes" del sistema en dispositivos compatibles.

Caracteristicas

- Android 7.0 Nougat Cyanogenmod 14.0
- Kernel 3.10.9
- OpenGL ES 1.1/2.0/3.0 (aceleración GPU)
- OpenCL 1.1 EP (aceleración GPU)
- Función multiusuario activada (hasta 8 usuarios)
- Soporte para Ethernet integrado y Ethernet Gigabit USB 3.0 externo
- Soporte para dongle Ralink Wireless USB, RTL8188CUS y RTL8191SU
- Soporte para dongle USB GPS
- Anclaje USB
- Punto de acceso Wi-Fi portátil
- Soporte DAC USB nativo Android
- Soporte para Webcam USB UVC
- Soporte para HDMI-CEC
- Selinux Enforce
- Bluetooth y USB-3G no funcionan todavía

Instalación

Para instalar Android Nougat en un soporte vacio, necesitas preparar tu tarjeta SD o eMMC con la correspondiente imagen de auto-instalación, que está disponible en http://bit.ly/2eWb1zi. Escribe la imagen en tu tarjeta SD o eMMC usando Win32DiskImager (http://bit.ly/1Vk9u4o).

Si estás actualizando desde la imagen CM 13.0, copia y pega la siguiente URL en la sección URL de ODROID-Updater:

http://oph.mdrjr.net/voodik/5422/ODROID-XU3/Android/CM-14.0/Alpha-0.1_10.10.16/update.zip

El proceso de actualización puede demorarse hasta 20 minutos, así que ten paciencia.

Figura 1 - Captura de Android Nougat

Conejos prácticos

Para que el Wifi funcione configura el nombre de módulo correcto en el archivo build.prop. Por ejemplo, para usar un Realtek 8192cu (por defecto), usa el siguiente parámetro:

```
wlan.modname=8192cu
```

Para usar Realtek 8188eu:

```
wlan.modname=8188eu
```

Para usar Ralink RT33XX/RT35XX/RT53XX/RT55XX:

```
wlan.modname=rt2800usb
```

Si estás utilizando un dongle USB GPS, configura el puerto tty correcto y la velocidad en build.prop:

```
ro.kernel.android.gps=ttyACM0
ro.kernel.android.gps.speed=9600
```

Para activar la posibilidad de apagar el sistema sin confirmar pulsando el botón de encendido, agrega la siguiente línea:

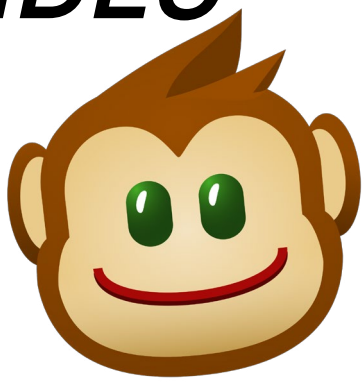
```
setprop persist.pwbtn.shutdown
true
```

Si deseas activar la función de impresión en la nube, descarga e instala la aplicación Google Cloud Print Service desde http://bit.ly/2e2YyLw.

Para comentarios, preguntas o sugerencias, visita el post original en http://bit.ly/2ec67zI.

REPRODUCCION DE VIDEO ACELERADA PARA NAVEGAR CON EL C2

EL CONTENIDO DE TUS SITIOS WEB EN FULL HD



por Adrian Popa

Si estás utilizando tu ODROID-C2 como ordenador de escritorio, lo más probable es que también lo uses para navegar por la web. A estas alturas, es posible que hayas notado que la reproducción de vídeo en el navegador no tiene el rendimiento que esperarías. Aunque puedes ver videos 720p de Youtube en el navegador, suelen entrecortarse, de modo que la única resolución aceptable para poder reproducir videos correctamente es 360p. Otros sitios simplemente informan que no pueden reproducir los videos.

Para controlar la calidad de reproducción de los vídeo se necesitan dos cosas: Decodificación de vídeo acelerada, que es gestionada por el paquete aml-libs en el C1/C2 y renderizado acelerado, del cual se encarga los controladores X11. Desafortunadamente, Chrome no soporta la decodificación acelerada de vídeo para aml-libs, de modo que la reproducción se realiza con ayuda de la CPU, dando lugar a este tipo de reproducciones lentas y entrecortadas en HD. El objetivo de este artículo es redireccionar la reproducción de vídeo a un proceso que pueda hacerlo utilizando aceleración por hardware, mejorando así la reproducción de vídeos en el navegador.

El mejor método

La mejor forma de hacer esto es, en primer lugar, recomponer el huevo antes de que tengamos problemas con el pollo. Con esto, nos referimos a añadir soporte aml-libs al código fuente de Chrome. Si observas la documentación de Chromium (http://bit.ly/2eTfcww), puedes ver que se utiliza un paquete ffmpeg personalizado para la decodificación de vídeo. Esto significa que tan pronto como algunos desarrolladores exporten el soporte aml-libs a ffmpeg, podríamos tener compilaciones de Chromium que reproduzcan videos por aceleración en C1/C2. Los parches para ffmpeg creados por el usuario del foro @LongChair deberían permitir la reproducción nativa al fin y al cabo (http://bit.ly/2eWjRwY). Con suerte, para cuando leas este artículo, esta técnica ya funcionará.

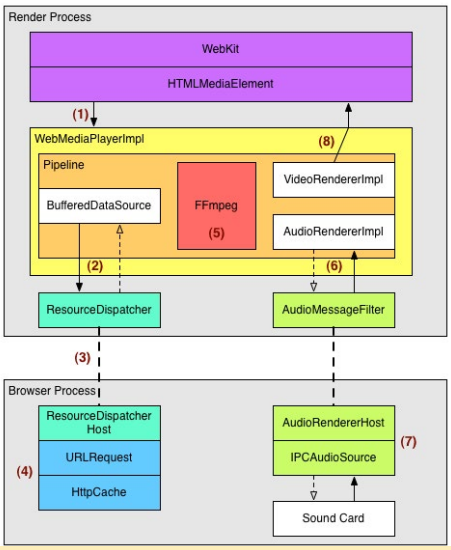


Figura 1 - Un diagrama del proceso de renderizado

El método enrevesado

En el caso de que el "mejor método" no esté disponible aún, tendremos que probar otra cosa. Esta técnica está inspirada en el trabajo realizado por un amigo en Firefox. Hay que tener en cuenta que podría no ser directamente aplicable a Chrome debido a cambios en el plugin API de Chromium (y su rechazo de NAPI). Así que, la cuestión es que mi amigo tiene un PC muy antiguo que utiliza para navegar por la web. Desafortunadamente, el PC no podía reproducir videos con el navegador, pero si podía hacerlo con un reproductor externo, de modo que mi amigo, cuyo nombre es Silviu, decidió hacer algo al respecto en lugar gastar dinero en el problema. Utilizó Firefox y el plugin Greasemonkey para escribir su propio mini-plugin para Firefox y se la ingenió para redireccionar el vídeo hacia el software reproductor de video mpv. Todo esto al mismo tiempo que conseguía que apareciera todo en la propia ventana del navegador dando así una impresión más coherente. Esto es lo que yo quería al principio, intentar reproducir en los ODROIDs.

Usó Greasemonkey para cargar una función JavaScript personalizada de 1600 líneas que anula todos los elementos de video HTML5 y los reemplaza con objetos proxy que reenviaran las llamadas a su plugin. Para ello necesitaba reimplementar toda la API de video HTML5 que se describe en https://mzl.la/2dRRTjP y https://mzl.la/2eTgUxS. Su código recibe las llamadas que hace el navegador para crear nuevos objetos de vídeo, configura la fuente de vídeo y permite iniciar, buscar y detener la reproducción.

Al interceptar los mensajes, el código JavaScript crea una solicitud con un tipo de contenido falso que es gestionado por una librería plugin diseñada específica-

mente para servir de intermediario en este escenario. El código JavaScript no tiene permiso para leer o ejecutar archivos, pero los plugins si, de modo que el plugin se utiliza para iniciar una instancia mpv y comunicarse con ella a través de una cone-xión redireccionada. El código JavaScript envía las solicitudes de reproducción, pausa y búsqueda al plugin, luego éste las convierte en comandos mpv y permite que el proceso mpv haga todo el trabajo pesado. Además, cuando controlas mpv a través de controles integrados, éste informa de cosas como son los eventos de reproducción/pausa/búsqueda a la librería del plugin, que los traduce de nuevo a eventos de JavaScript que se pasan a través del plugin Greasemonkey al reproductor original.

El resultado de todo esto es que el reproductor original muestra el estado correcto, la posición actual y se puede utilizar para controlar el reproductor de una forma intuitiva. Esto es importante porque muchos sitios tienen protecciones integradas contra el scraping y pueden realizar consultas periódicamente a través de JavaScript para ver el estado del reproductor y comprobar si algo extraño está sucediendo.

A nivel de presentación, usar un plugin te permite crear una nueva ventana del sistema operativo sin decoraciones en la barra de título y colocarla sobre el objeto de vídeo HTML5. Lo divertido es que la ventana está integrada dentro de la página web para que no sea visible mientas te desplazas por la pantalla. MPV puede utilizar el parámetro --wid para extraer su resultado a una ventana arbitraria, que es una opción interesante que funciona tanto en Windows como en Linux. Esto hace que el reproductor parezca que sea nativo y sencillo. Además, el plugin te permite crear multiples instancias del reproductor, de modo que puedes reproducir videos en varias pestañas simultáneamente.

Figura 2 - El MPV se reproduce en una ventana de Mousepad

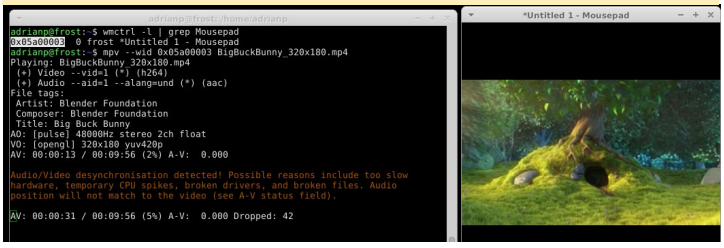
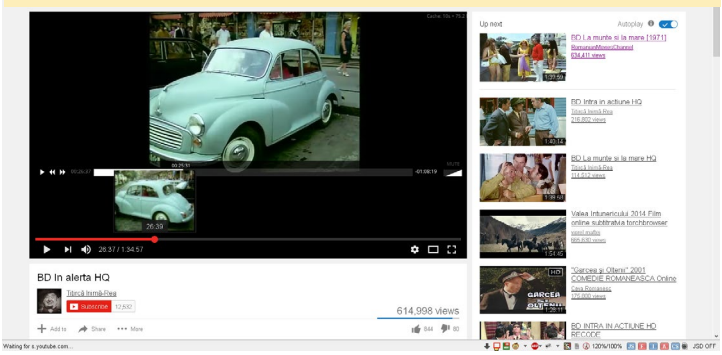


Figura 3 - Prueba de Firefox con controles MPV



Desafortunadamente, la solución que he visto sólo fue probada en Firefox y Windows, pero con un poco de trabajo, podría exportarse a Chromium. Tengo que convencer a mi amigo para que libere sus fuentes para que otros puedan participar.

La solución resultante

Una forma relativamente simple de obtener reproducción acelerada desde el navegador es pasar la URL del video a un reproductor que tenga soporte para aml-libs. Afortunadamente, tras una larga y muy entretenida discusión en el foro en http://bit.ly/2eLzOpA, ahora contamos con un reproductor independiente para el C2, así como parches para ffmpeg y mpv que podrían dar lugar a una selección más diversa de reproductores en el futuro. Sin embargo, para nuestras necesidades, usaremos c2play, un reproductor minimalista desarrollado específicamente para el C2 por el usuario del foro @crashoverride (http://bit.ly/2eXOp0s).

Nuestro proyecto consiste en lo siguiente: 1) crear un plugin chrome para enviar la URL actual (o la URL del elemento video) a un script backend, y 2) el script backend llama a youtube-dl para obtener la URL del video (si es necesario) y activa el reproductor para que lo reproduzca.

Afortunadamente, Chrome tiene una API de mensajería nativa (http://bit.ly/1cOVcrU) que te permite comunicarse con procesos externos (que es lo suficientemente buena como para utilizarla), así que esto es lo que haremos.

Para instalar c2play, necesitarás consultar las últimas instrucciones del hilo de soporte en http://bit.ly/2eXOp0s, ya que su desarrollo está avanzando muy rápido. Puedes instalarlo con los siguientes comandos:

```
$ git clone -b beta1 https://github.com/OtherCrash-
Override/c2play.git
$ sudo apt-get install libasound2-dev \
  libavformat-dev libass-dev libx11-dev premake4
$ cd c2play
$ premake4 gmake
$ make c2play-x11
$ sudo cp c2play-x11 \
  /usr/local/bin/c2play-x11
```

También necesitarás youtube-dl, que lo puedes obtener de los repositorios de Ubuntu, o desde http://bit.ly/1Vej3Vi. Youtube-dl es un programa que toma una URL de una página y extrae las URL de los elementos de vídeo de su interior. Puedes usarlo para una amplia gama de sitios web que utilizan HTML5 o Flash para la reproducción (http: // bit.ly/2d9yknp). Te recomiendo que lo instale manualmente, porque los sitios web sufren cambios muy a menudo y necesitarás actualizarlo fácilmente, lo cual se puede hacer con el siguiente conjunto de comandos:

```
$ sudo curl -L \
  https://yt-dl.org/downloads/latest/youtube-dl \
  -o /usr/local/bin/youtube-dl
$ sudo chmod a+rx \
  /usr/local/bin/youtube-dl
```

Por último, para instalar el plugin de Chrome que vincula todo esto, sigue estos pasos:

```
$ git clone https://github.com/mad-ady/odroid.
c2.video.helper.git
$ cd odroid.c2.video.helper
$ sudo apt-get install libjson-perl \
  libproc-background-perl libconfig-simple-perl
```

Para instalar el plugin, necesitarás navegar dentro de Chrome a chrome://extensions/. A continuación, arrastra y suelta desde un explorador de archivos la extensión ubicada en odroid.c2.video.helper/release/odroid.c2.video.helper-1.1.crx (o la versión 1.2, como se describe a continuación) en la ventana de Chrome, y se te pedirá que la instale.

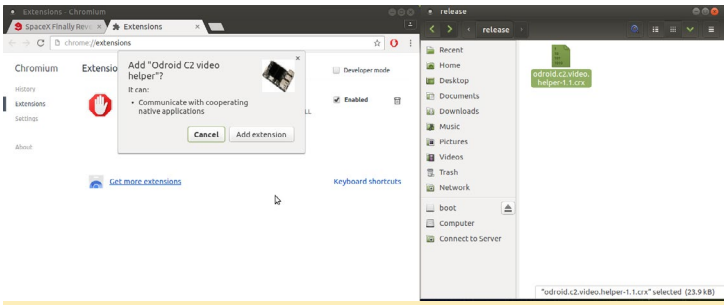


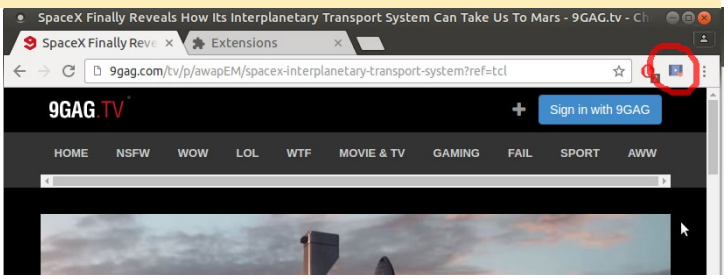
Figura 4 - Instalando nuestro plugin

Una vez instalado, continua instalando el script backend:

```
$ cd host
$ bash install_host.sh
```

El script install_host.sh crea los directorios necesarios en tu instalación de Chromium y copia el script backend (generalmente a $HOME/.config/chromium/NativeMessagingHosts) y su archivo de configuración a tu directorio personal. Llegados a este punto, deberías ver un nuevo botón que se parece a un reproductor de video junto a tu barra de direcciones. Si puedes

Figura 5 - ¡Pulsa el botón!



crear un ícono mejor, por favor, haz tu contribución al hilo de soporte al que se hace referencia al final del artículo.

Tras activar el plugin, cuando navegues a un sitio web que tenga un video integrado, puedes presionar este botón y la URL pasará al script backend. Este lo ejecutará a través de youtube-dl, que puede llevarle unos 5-6 segundos y obtendrá la URL del video. A continuación, llamará al reproductor con esta URL y la reproducción comenzará. Tienes un video del proceso de instalación y reproducción en https://youtu.be/Z3ng8Qm8STI.

Sólo lo he probado con c2play-x11 con una reproducción a pantalla completa. Para controlarlo, consulta la página de ayuda en http://bit.ly/2eWl5IV. El soporte para reproducción en ventanas se encuentra en la lista de tareas de @crashoverride (http://bit.ly/2fd8C7a) y probablemente se convertirá en una realidad en un futuro muy cercano. Además, la migración mpv de @LongChair podría utilizarse para la reproducción en el futuro. Por ahora con el mpv estándar la reproducción va bien para el contenido SD, pero a 720p aparece retardo. Puede ver una demostración en https://youtu.be/fJLv0cwPyfg.

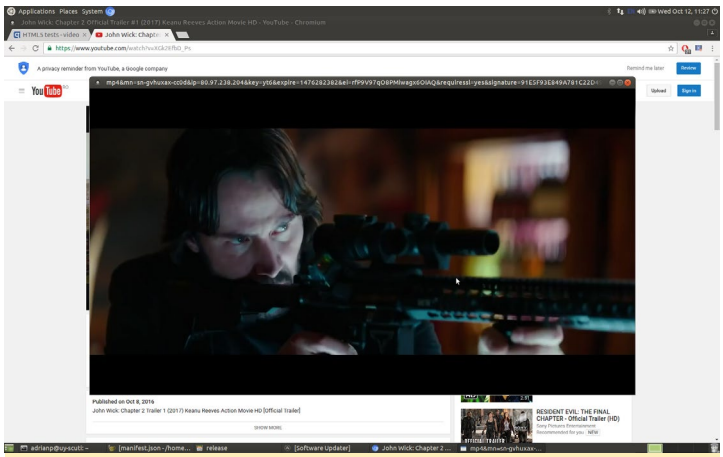


Figura 6 - Reproducción sin aceleración en mpv

El script backend tiene un archivo de configuración almacenado en ~ / .odroid.c2.video.helper.conf que usa una sintaxis ini. Con este archivo, puede desactivar la depuración, activada por defecto. También puede cambiar el reproductor y configurar algunos parámetros como la calidad, para varios sitios compatibles con youtube-dl. Por ejemplo, el archivo de configuración por defecto ajusta la calidad de youtube a 720p (-f 22) y desactiva la compatibilidad con las listas de reproducción. Puedes agregar nuevas secciones, que deben tener un nombre que coincida con el dominio de la URL. Además, el nombre de la sección no debe contener puntos (".") ya que no los soportan el módulo analizador de configuración. Para ver la información de depuración puede ejecutar:

```
$ sudo journalctl -f
```

Reproducir a 1080p o 4k desde YouTube es un poco más

```
adrianp@uy-scuti:~$ cat .odroid.c2.video.helper.conf
[general]
debug=1
playerdebug=1
player=/usr/local/bin/c2play-x11

[youtu]
;set quality to 720p
extraArgs="-f 22 --no-playlist"
;set quality to 3840x2160 with audio mp4a.40.2@128k (44100Hz)
;extraArgs="-f266,140 --no-playlist"

[vimeo]


adrianp@uy-scuti:~$
```

Figura 7 - Una muestra de configuración

problemático, ya que para resoluciones superiores a 720p, YouTube divide el vídeo y el audio en dos flujos distintos y su reproductor los mezcla en el navegador del cliente. Esto no sucede en otros sitios como Vimeo. Afortunadamente, @crashoverride también tiene una versión llamada dualstream que intenta soportar esto, como se describe en http://bit.ly/2fd7RLt. Las pruebas preliminares muestran que va por el buen camino. Logre reproducir video 4K con su reproductor, pero la reproducción por desgracia, se detuvo a los pocos segundos. Posiblemente mejore en un futuro cercano.

¿Qué funciona y qué no? Los sitios que soportan youtube-dl como YouTube, Vimeo, Facebook, IMDB, Engadget, DailyMotion, TED, Cracked, Apple Trailers, 9gag TV y varios sitios para adultos deberían funcionar sin problema. Para reproducir contenido, debes dirigirte a una página con un único video y hacer clic en el botón situado junto a la barra de direcciones. Sin embargo, los sitios que requieren un login para ver el contenido no funcionarán, ni los sitios que requieran cookies para acceder a los datos, debido a cómo se conecta el plugin y procesa los datos.

Valerse de youtube-dl para los sitios que son compatibles está bien, pero mi idea era recuperar la URL del video directamente desde el DOM del navegador. En caso de que no sepas cómo funcionan las cosas en Internet, las páginas web pueden crear objetos de vídeo que tienen varias propiedades, incluyendo una fuente que apunta al video que se está reproduciendo. Youtube-dl analiza la página y con mucha magia interna consigue extraer esta URL y la presenta a otros procesos. Sin embargo, ya que estamos dentro del navegador, deberíamos ser capaces de obtener la fuente del video, ahorrando así cerca de 6 segundos de tiempo de procesamiento. Modifiqué la extensión de Chrome para buscar objetos de video y mi intención era crear un botón que iniciara la reproducción externamente. Pero esto estaba seguro que fallaría, ya que cada página incluye el vídeo bajo varias capas de objetos, la mayoría de los cuales están ocultas o se superponen con otras, de modo que el botón se oculta o aparece incorrectamente en la página.

Mi siguiente intento fue interceptar el evento "play" y pausar el video, coger la URL y pasarla al backend para su reproducción. Esto funcionó sorprendentemente bien en mi sitio de pruebas (http://bit.ly/1nAXG0z), pero no para sitios más grandes como Youmerges

Figura 8 - Objeto de video normal vs fuente de medios

```
    crossOrigin: null
    currentSrc: "http://www.quirksmode.org/html5/videos/big_buck_bunny.mp4"
    currentTime: 12.766221
```

```
crossOrigin: null
currentSrc: "blob:https://www.youtube.com/1155dccf-f07e-4967-86de-6e5ded2ad7a8"
currentTime: 3.889002
```

Tube y Vimeo. El análisis de las páginas reveló el motivo: estos sitios usan una técnica llamada "media-source" (http://bit.ly/2eWmfnG) que utiliza JavaScript para hacer peticiones y retransmitir el video, de modo que no podemos extraer una URL para pasarla a la reproducción.

Para los sitios que utilizan un elemento de vídeo con una URL real como fuente, el plugin reproducirá el contenido con el reproductor externo en el primer intento. Si intentas reproducir el vídeo de nuevo sin recargar la página, se reproducirá normalmente dentro de la página. Se trata de una protección en caso de que el reproductor externo no pueda manejar el formato de vídeo y quieras volver al navegador. En este caso, sólo deberías ver una ventana negra parpadeando por un instante mientras se intenta reproducir, de lo contrario se reproducirá con normalidad.

He lanzado dos versiones del plugin. La versión 1.1 soporta sólo la reproducción a través de youtube-dl usando el botón de la barra de herramientas. La versión 1.2 añade soporte experimental de reproducción directa, pero no siempre funciona, o puede ser demasiado molesto para los videos que se reproducen automáticamente al entrar en la página, de modo que el plugin que utilices dependerá de tus necesidades

Me gustaría escuchar tus sugerencias y comentarios en el hilo de soporte en http://bit.ly/2eBVtCZ. Todo esto no habría sido posible sin el duro trabajo de @crashoverride y @LongChair, así que envíales tu agradecimiento.

CONOCIENDO UN ODROIDIAN

JOACHIM ALTHOF

editado por Rob Roy

Joachim en lo alto de una turbina de viento en Estonia durante un viaje de encargo en abril de 2015

Hablanos un poco sobre ti.

Mi nombre es Joachim Althof, y tengo 32 años. Vivo cerca de Hanover en Alemania junto con mi esposa Katrin y nuestras dos hijas. Realice mis estudios de mecánica y electrónica en la Universidad de Ciencias Aplicadas de Lemgo, que es una pequeña ciudad al noroeste de Alemania. Después de terminar mis estudios, trabajé para una empresa cerca de la frontera con Dinamarca, donde escribí un software microcontrolador integrado para invertidores de red relativamente grandes (> 100 kW) y otros tipos de convertidores fotovoltaicos y eólicos. Cinco años más tarde, volví a mi ciudad natal y empecé a trabajar para una nueva empresa. Soy el primero y hasta ahora el único desarrollador de software para pequeños invertidores PMSM (<10 kW), y soy responsable de muchas otras cosas relacionadas con la coordinación del desarrollo.

¿Cómo fueron tus inicios con los ordenadores?

Experimente mis primeras experiencias con ordenadores con el Commodore64 de mi padre a principios de los 90. Pensando en ello, en realidad era de la vieja escuela, con un monitor de diferentes tonalidades que parpadeaba mucho, un botón de reinicio y un altavoz casero, junto con una impresora de puerto paralelo de 9 pines con "papel ilimitado" gris. Al principio, no tenía ni idea de lo que ocurría cuando escribía <load "$", 8,1>. Y honestamente, realmente no me importaba, siempre y cuando los juegos se iniciaran. Algo más tarde, empecé a interesarme en entender lo que sucedía dentro de esa caja gris. Incluso logré ejecutar GEOS, que era un tipo de sistema operativo gráfico para el C64, y conseguí instalar el controlador de impresora. A los 12 años, conseguí mi primer ordenador i486 que ejecutaba DOS. Debido al rápido desarrollo del hardware en ese momento, fui actualizando este dispositivo poco a poco pieza por pieza y se convirtió en mi primer ordenador "serio" durante muchos años. En casa, mi esposa dice que soy "el responsable de todo lo que tiene cables".

¿Qué te atrajo de la plataforma ODROID?

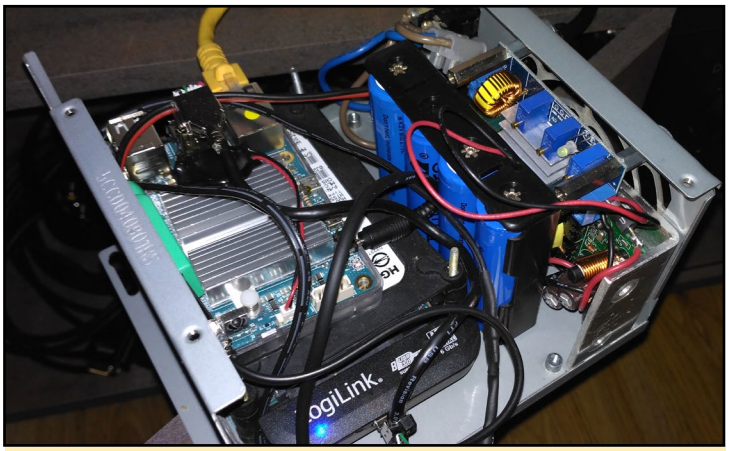
Cuando empecé con los SBCs, la Raspberry Pi ya estaba muy extendida. Quería unirme a la novedad, pero deseaba algo diferente y más potente que la RPi. Decidí comprar un Cubieboard2, que era lo más novedoso por aquellos entonces. Aprendí mucho y conocí a gente realmente agradable en el foro. Sin embargo, pronto me decepcionó un poco la compañía, la comunidad y el producto, así que busqué una nueva plataforma para jugar, y así es cómo descubrí la plataforma ODROID. Me impresionaron las especificaciones de hardware y el precio era razonable, así que compré mi primer ODROID, el modelo C1. Exporté todos mis proyectos a éste y fue divertido ver lo que podía hacer, también descubrir todas sus peculiaridades.

Estaba muy satisfecho con el rendimiento, porque hizo todo lo que quería que hiciera. Cuando salió el C2, le dije a mi esposa: "¡Mira esto! ¡Necesitamos este! ¡Más es mejor!". Es cierto que a ella no le importaba mucho, pero compré mi primer C2 y un poco más tarde, mi segundo C2 llegó, acompañado de mi esposa agitando su cabeza hacia mí.

He descubierto que la comunidad ODROID es muy activa. Por supuesto, hay miembros que son más activos que otros, pero en general, me sorprendió gratamente la comunidad. Lo que aprecio mucho es el hecho de que incluso los administradores y los miembros de Hardkernel contribuyan al foro. Para mí, parece que hay una colaboración activa entre los miembros de Hardkernel y la comunidad de Internet, que no tiene precio. Una placa no tiene casi nada de valor sin el soporte adecuado.

¿Cómo usas tus ODROID?

Mi primer planteamiento era conseguir que todos mis proyectos funcionaran en una única placa. Mi C1 inicial era un servidor de archivos en red, un servidor de medios, un reproductor multimedia, un servidor SVN, una máquina de juegos, un sistema Ambilight, una placa de desarrollo y un cañón de defensa orbital. ¡En Alemania, lo llamamos una "egg-laying woolmilk-sow"! Sin embargo, cada vez que jugueteaba un poco con él, se fastidiaban varias cosas, lo que provocó un bajo Wife Acceptance Factor (WAF). Decidí descentralizar mi sistema, lo cual se tradujo en la necesidad de más ODROIDs, lo que llevó a mi esposa a decir: "Espera, ¿para qué es esto?" Ahora tengo mi C1 como un servidor dedicado, que será exportado a un V2 con el tiempo, un C2 como reproductor multimedia que ejecuta LibreELEC, y un C2 con el que jugar. Más adelante usaré uno de los ODROIDs como plataforma de juegos retro.



El servidor de casa de Joachim alimentado por un C1, todo bien montado en una vieja carcasa de una fuente de alimentación ATX

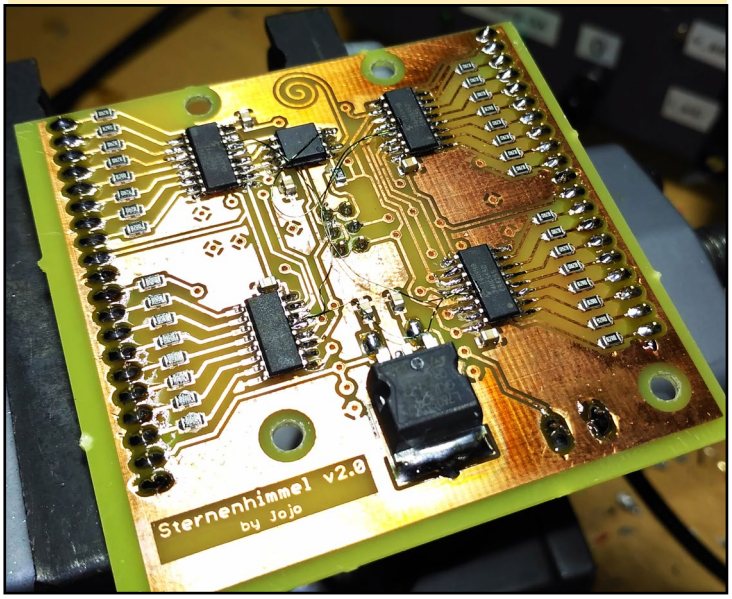
¿Cuál es tu ODROID favorito y por qué?

Por ahora, sólo tengo experiencia con el C1 y C2, aunque mi favorito es el C2. Es muy potente para su precio, es pequeño, versátil y cuenta con un buen soporte. Ambas placas funcionan muy bien en los proyectos donde las he utilizado. Creo que todavía hay muchos más proyectos y posibilidades, una vez que la arquitectura ARM de 64 bits se vuelva más madura. También me gusta el cabezal E/S del C1 y el C2, ya que facilita los pequeños ajuste en hardware. Hacer que este cabezal de E/S compatible con la Raspberry Pi es otra historia.

¿Qué innovaciones te gustaría ver en futuros productos de Hardkernel?

Esta es una muy buena pregunta, porque muchas características ya están incluidas. Me gustaría tener wifi integrado o tal vez USB 3.0, que ya está disponible en la XU4. Lo más importante para mí es disponer de buen soporte, ya que pase dificultades con algunos drivers USB en el C1. Además, algo similar al RPi Zero sería genial. Sé que Hardkernel ofrece el C0, pero éste no es comparable con el RPi Cero. Creo que alguna gente podría usar también un puerto MIPI-CSI, especialmente si fuera compatible con la RPi. Esto podría abrir un nuevo y enorme campo para las aplicaciones ODROID.

Una PCB casera para un PWM de 32 canales, alimentado por un ATTiny85 a través de SPI y un montón de registros para montar una lámpara de techo en la habitación de su hija para que parezca un cielo nocturno lleno de estrellas brillantes.



¿Qué aficiones e intereses tienes aparte de los ordenadores?

Me gusta hacer deportes como correr y Freeletics. Soy un fan de la música metal y toco la batería. Mi voz tampoco es mala, y me gusta ir a conciertos de música y festivales. Durante los días oscuros y fríos del invierno, trabajo en un proyecto integrado con microcontroladores AVR y una gran cantidad de soldaduras. Lo más importante, disfruto no haciendo nada así como también pasar tiempo libre con amigos y mi familia.

¿Qué consejo la darías a alguien que quiere aprender más sobre programación?

Depende del tipo de programación específica. Creo que aprender C incrustado es una forma difícil de empezar, ya que es menos "visual" que escribir software para un PC. Sólo puedes ver o medir los resultados de tu programación con E/S de hardware. Sin embargo, vale la pena el esfuerzo si piensas en las posibilidades. Empieza con algo simple como un bucle LED parpadeante. No te asustes con los punteros *, aunque no son necesarios para los programadores principiantes. Ten en cuenta siempre lo poderoso que puede llegar a ser el software. El software es el alma de un dispositivo, ¡y dar vida a una pieza de hardware muerta puede ser impresionante! Además, saber que algo puede explotar al fijar un bit incorrecto puede ser muy emocionante y aterrador al mismo tiempo. Por otro lado, la programación de scripts shell o C # puede ser buena para empezar, ya que es más visual que la programación incrustada. Realmente depende de la aplicación en sí. Lo más importante: ¡simplemente hazlo! Muestra algo de auto-iniciativa y la gente te ayudará a medida que te vayas ayudando a ti mismo.

Joachim y sus chicas disfrutando en la feria, aunque ellas disfrutaron más de lo que lo hizo él